[redacted]

UIUCDCS-R-75-732

PROGRAM MANUAL:
NOR NETWORK TRANSDUCTION BASED ON ERROR-COMPENSATION
(Reference Manual of NOR Network Transduction
Programs NETTRA-E1, NETTRA-E2, and NETTRA-E3)

BY

H. C. Lai
J. N. Culliney

June, 1975

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-75-732

PROGRAM MANUAL:
NOR NETWORK TRANSDUCTION BASED ON ERROR-COMPENSATION
(Reference Manual of NOR Network Transduction Programs NETTRA-E1, NETTRA-E2, and NETTRA-E3)

BY

H. C. Lai
J. N. Culliney

June, 1975

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

This work was supported in part by the National Science Foundation under Grant No. GJ-40221.

ABSTRACT

Three NOR network transduction procedures based on error-compensation were implemented in the FORTRAN computer programs NETTRA-E1, NETTRA-E2, and NETTRA-E3. The general principles on which these programs are based are discussed in a separate report. The present report, however, describes the specific implementations of the three programs and serves as a reference manual for the program user. Preparation of input data is discussed in detail.

Transduction (trans formation and reduction) procedures attempt to reduce given, non-optimal, multiple-output, multiple-level, loop-free, NOR-gate networks to "near-optimal" networks of fewer gates. The three programs described in this report, based on the sophisticated "error-compensation" concept, remove gates one at a time from the network and, after each removal, try to reconfigure the network, without employing additional gates, to compensate for any resultant errors caused in the network output(s).

TABLE OF CONTENTS

1. INTRODUCTION

This manual is intended to instruct the reader in the use of the FORTRAN programs 'NETTRA-E1,' 'NETTRA-E2,' and 'NETTRA-E3,' and also to enable a moderately detailed understanding of how these programs actually realize their respective algorithms. The principal algorithm upon which these programs are based is described in detail in [5], and this manual will assume a knowledge of the definitions and algorithm descriptions in [4] and [5].

NETTRA-E1, -E2, and -E3 represent only three out of a whole system of programs developed at the University of Illinois by the logical design group of S. Muroga. The generic name 'NETTRA' (for NETwork TRAnsformation) designates the whole collection of programs comprising the system. All of the programs in the NETTRA system either transform or assist in transforming networks of interconnected NOR gates realizing various functions (either completely or incompletely specified) of their respective sets of input variables. By these transformations, a large, non-optimal network of NOR gates realizing one or more various functions can often be reduced to a smaller, less expensive (in terms of the number of required gates and interconnections, for example), near-optimal network realizing the same function(s). In general, such a transduction (transformation and reduction) could involve a complete reorganization of the network: the addition and/or deletion of gates; the addition and/or deletion of connections among gates;

and/or the substitution of certain connections for various others. The transduction procedure realized by NETTRA-E1 (the same procedure forms the basis of NETTRA-E2 and -E3 also) can accomplish any of these changes, with the exception of adding gates to the network.

The present three programs employ a transduction procedure much more powerful than those found in NETTRA-P1, -P2, -PG1, -G1, -G2, -G3, and -G4 (see [1], [2], [3], [6], [7], [8]). The transduction procedures embodied in these earlier programs never make a change to a network which would alter its output functions. However, the current procedure is able to go through a long series of networks representing intermediate stages of the transformation, none of which realize the correct output function(s), in order to finally obtain a less expensive network correctly realizing the desired outputs. In this sense, this procedure is more "far-sighted" than the transduction procedures realized by the earlier programs. Although the feature is not exploited in the programs explained here, it is possible for the current procedure to use an initial network which does not even realize the desired function(s) (hopefully, though, it does realize a function "reasonably close" to the desired one).

NETTRA-E1, -E2, and -E3 are primarily intended to reduce the number of gates in a given network. No serious attempts are made by these programs to minimize the number of connections. However, other transformation programs (e.g., NETTRA-P1, -P2, or -G1) can be applied after NETTRA-E1 (or -E2 or -E3) to try to further reduce the number of connections in a network.

The following section, Section 2, explains the NETTRA-E1 program which applies just once the transduction procedure discussed in [5] to a given network. This single application attempts to eliminate just a single gate from the network. In order to eliminate several gates, the procedure must be applied several times. Two different methods of doing this (corresponding to the programs NETTRA-E2 and NETTRA-E3) are discussed in Section 3. Section 4 briefly describes all of the subroutines used in the programs NETTRA-E1, -E2 and -E3. In Section 5, instructions are given on the preparation of the input data for the three programs. Finally, a listing of all of the FORTRAN subroutines used in NETTRA-E1, -E2, and -E3 is given in the appendix.

2. ERROR-COMPENSATION PROCEDURE

In this section, the NOR-network transduction procedure realized by the FORTRAN program designated NETTRA-E1 is discussed. When it is applied in an attempt to transform a network, the number of gates in the network will either be reduced or left unchanged - it will never be increased.

In contrast to the earlier programs (NETTRA-P1, -P2, -PG1, -G1, -G2, -G3, and -G4) which never transform a network so that it produces incorrect output functions, NETTRA-E1 causes "errors" to appear in the outputs of a network by deliberately removing a necessary (to the correct operation of the network) gate of the network. Then it attempts to compensate for these errors by adding and rearranging connections in the remaining network.

Actually, NETTRA-E1 "memorizes" the original network and removes each of its gates in turn, trying to compensate for the errors in the new networks which each have one less gate than the original.

If the program is successful in compensating for any of these removed gates, it prints out the solution (i.e., the transformed, reduced network of a smaller number of gates than the network originally given as input) and halts.

The input to this program and NETTRA-E2 and -E3 is a description of a particular NOR network under consideration. This description (explained in great detail in Section 5) consists of a set of variables and arrays containing various network parameters.

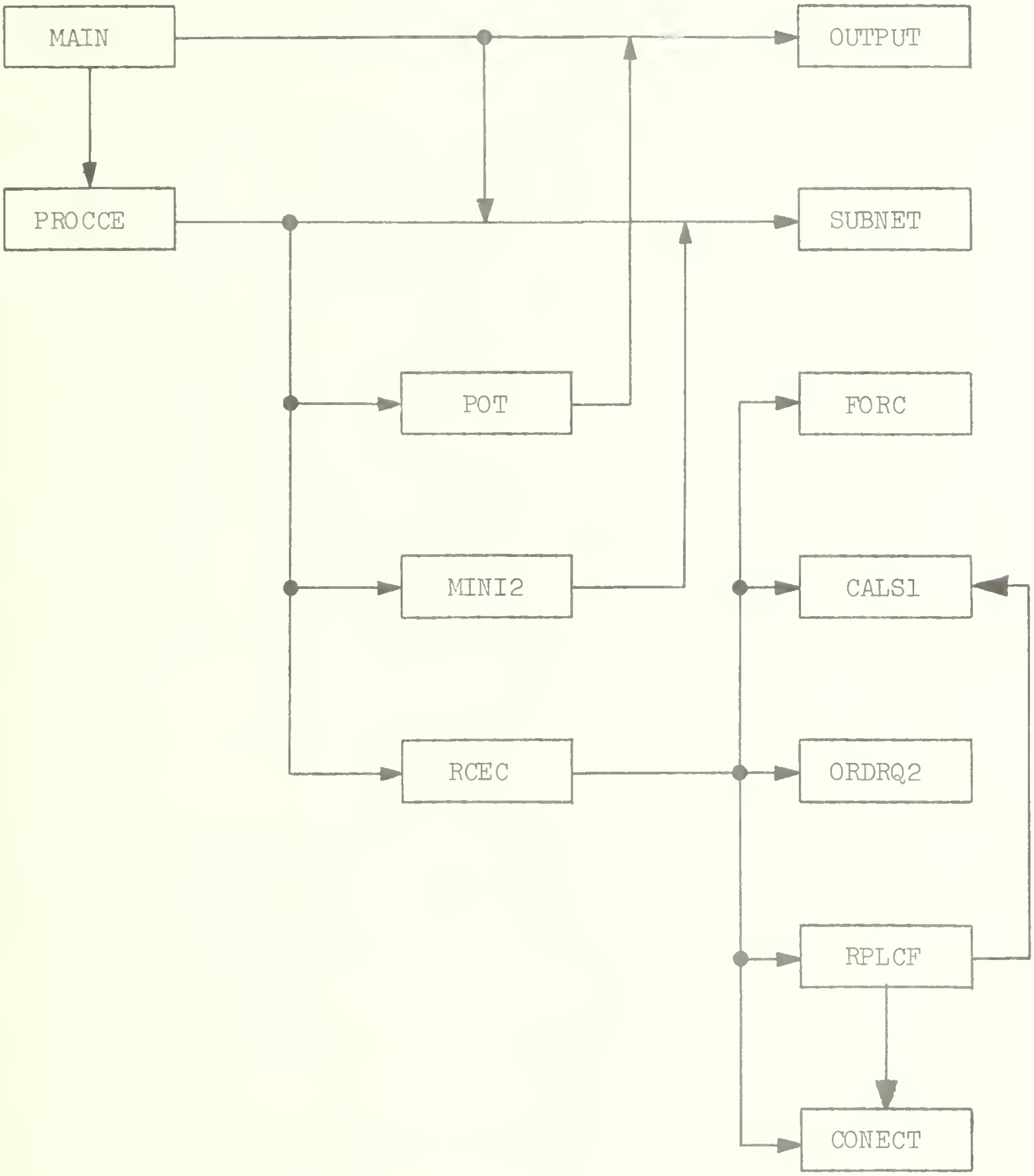


Figure 2.1 General organization of the programs NETTRA-E1 and NETTRA-E2.

The entire NETTRA-E1 program requires 163K bytes of core storage, about 78K being occupied by the actual program instructions and about 85K by the stored data.

The following subroutines, written in FORTRAN IV for the IBM 360/75, constitute the program NETTRA-E1: CALS1, CONECT, FORC, MAIN, MINI2, ORDRQ2, OUTPUT, POT, PROCCE, RCEC, RPLCF, and SUBNET. Two system-supplied timing routines, STIMEZ and KTIMEZ are also assumed to be available, but if they are not, their use can be omitted from the program, or another suitable timing routine substituted, without harming the procedure itself.

The general organization of the program NETTRA-E1 is shown in Figure 2.1. An arrow from block i to block j represents the fact that the subroutine represented by block i calls the subroutine represented by block j.

2.1 General Procedure and Flowchart

The general execution of the error-compensation procedure is carried out by the subroutine PROCCE (for: transduction PROCedure by the Compensation of Errors) which, while quite simple itself, controls the calling of the major subroutines (explained in more detail in Sections 2.2 and 2.3) that actually execute the complex details of the procedure. The following discussion of PROCCE will assume a knowledge of the information contained in [4] and [5].

Explanations of the purposes of the variables and arrays actually appearing in the subroutine can be found in the program listing of PROCCE in the appendix. It is, however, convenient to define some of the variables at this point in order to discuss the flowchart of PROCCE which appears in Figure 2.1.1:

N is the number of external variables, n, if only uncomplemented variables are allowed as inputs. If both complemented and uncomplemented are available (i.e., n variables and their n complements) then N is equal to 2n. Note that this is strictly the representation internal to the program; for input-output purposes (as described in Section 5) N and n are always equal.

R is the number of gates specified by the input data to the program. It includes all gates declared to be present by the input data, even though some of them may be isolated (i.e., not connected to other gates in the network). Internally, the program represents the gates 1, 2, ..., R by the labels N + 1, N + 2, ..., N + R. (External variables are labeled 1, 2, ..., N internally.)

NR is equal to the sum N + R. It is often convenient to treat both external variables and gates in a similar manner. External variables being labeled 1, 2, ..., N and gates being labeled N + 1, ..., N + R (internally), the number N + R is frequently required.

GSMALL is a two-dimensional array used to store intermediate and final calculated compatible sets.† GSMALL (i, j) contains (or rather will contain by the end of a procedure) the j-th component of the vector representing the compatible set of permissible functions for gate or external variable i.

† For simplicity, sometimes just the words "compatible sets" will be used to denote compatible sets of permissible functions.

NEPMAX is a variable limiting the maximum number of "error-positions" which will be tolerated in a network. NEPMAX is either specified by the input data, or, if left unspecified on the input cards, it is set to the value $2^{(n-1)}$ (for example, in the case of an implicit specification of external variables). An error-position is an index number i such that at least one output gate of a network (with errors) has an incorrect output for the i-th network input vector (i.e., the i-th combination of 0 and 1 assignments to the network inputs). The removal of a gate from the network, as occurs during the execution of the procedure, usually causes errors to appear in several positions. If the number of these error-positions is too great, the chance of compensating all of them is generally low. In the interest of efficiency, if the number of error-positions exceeds NEPMAX after the removal of a certain gate, PROCCE does not attempt to compensate for the errors. Instead, it restores the original network and moves on to remove another gate.

In addition to these variables which appear in the FORTRAN program itself, a few concepts from [4] and [5] should be recalled:

$G_c(v_i)$ was defined in [4], and $G_E(v_i)$ was defined in [5]. The expression $G_c(v_i)$ denotes a vector representing a compatible set of permissible functions (CSPF) of a gate or external variable v_i[†]. This concept

[†] In [5], the v_i represent input terminals and gates, the concept of input terminals being introduced for theoretical completeness. Since the distinction between input terminals and external variables is unnecessary for the purposes of this paper, the v_i are considered to represent external variables and gates.

was frequently used in programs realizing earlier procedures: NETTRA-PG1, NETTRA-G1, NETTRA-G3, and NETTRA-G4. For the error-compensation procedure, however, CSPF's must be extended to the concept of compatible sets of permissible functions with errors (CSPFE's) as defined in [5]. For each gate or external variable v_i, the corresponding CSPFE is denoted $G_E(v_i)$. The notation $G_c^{(d)}(v_i)$ or $G_E^{(d)}(v_i)$ refers to the d-th component of the respective corresponding vector.

Whereas the components of the CSPF vectors were only of three types: 0, 1, or * (don't-care), the components of the CSPFE vectors may be any of five (logical) types: *, 0, 1, $\underline{0}$ (a 0 error), or $\underline{1}$ (a 1 error).

Blocks 1 through 4 of the flowchart of PROCCE (Figure 2.1.1) perform some preliminary steps in preparation for the main part of the procedure realized in blocks 5 through 20.

Block 1 calls the subroutine MINI2 (described in detail in [7]). This serves two purposes. First of all, MINI2 realizes a "pruning" transduction procedure, and it may be able to quickly eliminate some unnecessary gates from the original network. Secondly, MINI2 will calculate CSPF vectors ($G_v(v_i)$'s) for all of the gates remaining in the network.

The information is examined in block 2 where the number of 1's in the CSPF vector of each gate is determined.

Block 3 creates an ordering of gates 1 through R based on an increasing number of 1's in their respective CSPF vectors. The ordering is stored in the array PORDER such that the gate stored in the location PORDER(1) has a minimum number of 1's in its CSPF vector and the gate stored in PORDER(R) has a maximum number of 1's. In a rough sense, the number of

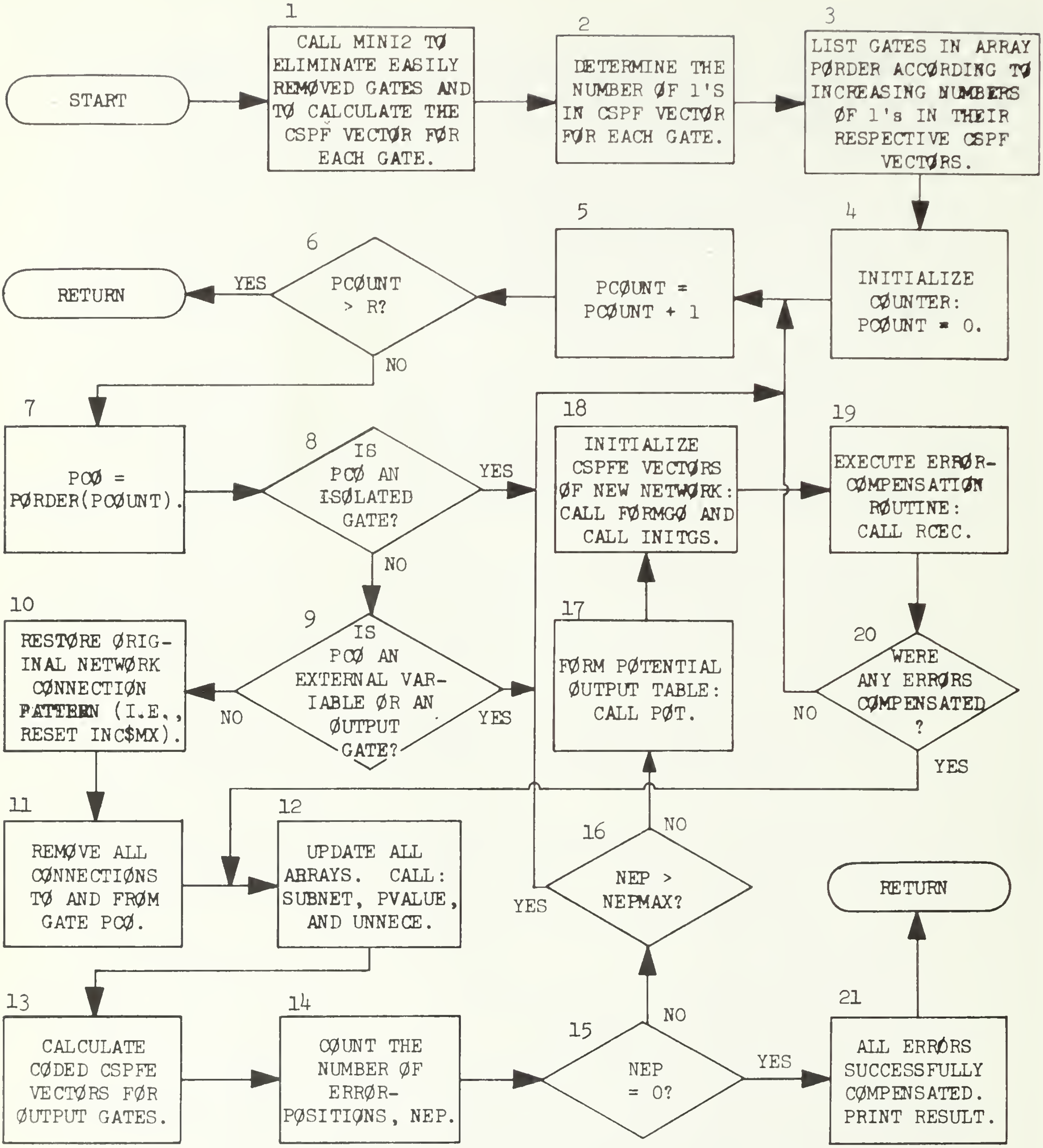

START
1
CALL MINI2 TØ ELIMINATE EASILY REMØVED GATES AND TØ CALCULATE THE CSPF VECTØR FØR EACH GATE.
2
DETERMINE THE NUMBER ØF 1'S IN CSPF VECTØR FØR EACH GATE.
3
LIST GATES IN ARRAY PØRDER ACCØRDING TØ INCREASING NUMBERS ØF 1's IN THEIR RESPECTIVE CSPF VECTØRS.
4
INITIALIZE CØUNTER: PCØUNT = 0.
5
PCØUNT = PCØUNT + 1
6
PCØUNT > R?
YES
RETURN
NO
7
PCØ = PØRDER(PCØUNT).
8
IS PCØ AN ISØLATED GATE?
YES
NO
9
IS PCØ AN EXTERNAL VARIABLE ØR AN ØUTPUT GATE?
NO
YES
10
RESTØRE ØRIGINAL NETWØRK CØNNECTIØN PATTERN (I.E., RESET INC$MX).
11
REMØVE ALL CØNNECTIØNS TØ AND FRØM GATE PCØ.
12
UPDATE ALL ARRAYS. CALL: SUBNET, PVALUE, AND UNNECE.
13
CALCULATE CØDED CSPFE VECTØRS FØR ØUTPUT GATES.
14
CØUNT THE NUMBER ØF ERRØR-PØSITIØNS, NEP.
15
NEP = 0?
YES
NO
16
NEP > NEPMAX?
YES
NO
17
FØRM PØTENTIAL ØUTPUT TABLE: CALL PØT.
18
INITIALIZE CSPFE VECTØRS ØF NEW NETWØRK: CALL FØRMGØ AND CALL INITGS.
19
EXECUTE ERRØR-CØMPENSATIØN RØUTINE: CALL RCEC.
20
WERE ANY ERRØRS CØMPENSATED ?
NO
YES
21
ALL ERRØRS SUCCESSFULLY CØMPENSATED. PRINT RESULT.
RETURN

Figure 2.1.1 Generalized flowchart of PRØCCE.

1's in a gate's CSPF vector reflects the relative "importance" of that gate in the network. In general the removal of a gate with many 1's in its CSPF vector from a given network is likely to produce more error-positions in the output gates than would the removal of a gate with fewer 1's. Thus, the ordering in the array PORDER roughly lists gates which are, from PORDER(1) to PORDER(R), increasingly more difficult to compensate for (the errors caused) when they are removed from the original network. Trying to remove the most easily compensated gates (i.e., those whose removals cause fewer error-positions) first (as is done in blocks 5 through 20) will generally lead to a quicker solution.

In block 4 a counter, PCOUNT, is initialized to the value 0. PCOUNT will be incremented by 1 during every pass through block 5. If PCOUNT exceeds the value R (i.e., the number of gates in the network), a condition tested for in block 6, it means PROCCE was unable to successfully compensate for the removal of any gate from the network, and PROCCE returns to the calling subroutine (MAIN).

If PCOUNT was found to be less than or equal to R, block 7 will set the variable PCO equal to PORDER(PCOUNT). PCO is then the label of the next gate to be removed from the network by the error-compensation procedure.

Block 8 checks to see if PCO is an isolated gate. If it is, there is, of course, no purpose in removing it from the network, and so the program would move on to the next gate to be removed (by returning to block 5).

If PCO is not isolated, block 9 further tests to see if PCO is an external variable (PROCCE assumes that all n variables are essential to the network) or an output gate. In either case, it cannot be removed from the network, and so the next gate is selected for removal instead (by returning to block 5).

If PCO passes these tests, PROCCE enters block 10. Here an array INC$MX is initialized to contain the connection pattern of the original network (i.e., the network connection pattern as it existed immediately after the transformation by MINI2 in block 1). The value INC$MX($v_i$, v_j) indicates the presence or absence of a connection from gate or external variable v_i to gate v_j.

This initialization is done in preparation for the removal of PCO from that original network. During the calculation, the array INC$MX always contains the most recently updated version of the network connection pattern.

Block 11 removes PCO from the orignial network by removing all of its input and output connections. This is done by changing some of the values of INC$MX.

The removal of PCO also causes changes in many other arrays storing various information related to the network configuration. These are updated by calling the subroutine SUBNET and its entry point PVALUE in block 12. Another entry point of SUBNET, UNNECE, is called to eliminate from the network any gates which may have been left, after the removal of PCO, with no paths to any of the network output gates. It is possible that leaving such gates in the network might actually be more beneficial, but an argument could be made either way. Programming considerations tipped the scale in favor of their removal.

Although there are only 5 logical types of components permitted in the CSPFE vectors, *, 0, 1, $\underline{0}$, and $\underline{1}$, the variety of the actual codings employed to realize these 5 types is somewhat larger (this will be discussed further in Section 2.2.1). In block 13 the outputs of all of the gates remaining in the new network are recalculated. The actual (new) network outputs are compared with the desired outputs, and from this information the (initial) coded entries of the CSPFE vectors of the network output gates can be directly determined.

The number of error-positions in this new network, NEP, are counted in block 14. If the number of error-positions is 0, the new network realizes all of the desired functions correctly, and block 15 directs control of the program to block 21.

Otherwise, the program enters block 16 where NEP is checked to see if it exceeds the maximum number of allowable error-positions, NEPMAX. NEPMAX is a parameter which may be varied by the user (see Section 5). If NEPMAX is exceeded, PROCCE abandons all hope of compensating for errors in so many positions, and control goes back to block 5 for the selection of a new PCO.

If NEPMAX was not exceeded, block 17 forms the "potential output table" by calling the subroutine POT (POT will be discussed in more detail in Section 2.2.2). Essentially, the potential output table lists all (theoretically all - but actually just a "great many") functions which either exist or can be "manufactured" (by adding the appropriate connections to the network) by a certain algorithm. This table is used to assist in error-compensation by providing a list of functions which may be connected as new inputs to certain gates during the procedure.

This is followed by another preparatory step just before the execution of the main error-compensation subroutine, RCEC. The CSPFE vectors are initialized in block 18 by calling INITGS (an entry point of MINI2). (The call to FORMGO merely recalculates GORDER in preparation for the execution of

INITGS.) This initialization sets the values of some components of certain CSPFE vectors which can be predetermined. This initialization generally improves both the efficiency and effectiveness of the procedure.

Block 19 is the most important part of the flowchart. Here, the heart of the procedure, subroutine RCEC, is called. The subroutine RCEC (discussed in detail in Section 2.3) performs a function somewhat similar to that of the subroutine PROCII in [1], except that it must also deal with the added difficulty of "errors" (i.e., undesired outputs of certain gates for certain input combinations) present in the network. The main purpose of RCEC is to rearrange the network connection pattern in a manner such that all of these "errors" can be compensated. This cannot be done in just a single call to RCEC however. When RCEC corrects just a single error of a certain type, it must return to PROCCE (block 12) to have all of the necessary arrays updated before it can continue its error-compensation task. If the correction of that error causes the network to function correctly, this fact will be detected in block 15.

If no errors of any kind were compensated by RCEC, it means that the compensation of further errors is impossible (at least, by following the algorithm realized by RCEC), so control of the program is sent back to block 5 to select a new PCO. The test is made in block 20.

When block 15 detects the presence of an error-free network (i.e., all of the desired output functions are correctly realized by the network) with the gate PCO removed, control goes to block 21 which prints out the new result. PROCCE then returns to the calling subroutine.

2.2 Major Support Subroutines

This section will discuss some of the details of two important support subroutines called by PROCCE, MINI2 and POT. A knowledge of the functions of these two subroutines is necessary in order to understand the error-compensation subroutine, RCEC (discussed in Section 2.3).

2.2.1 Subroutine MINI2

The subroutine MINI2 was discussed in an earlier report, [7], with an emphasis on those points of the subroutine most relevant to the operation of the program NETTRA-PG1. In block 1 of the flowchart of PROCCE (see Figure 2.1.1), MINI2 is used in a capacity similar to that in NETTRA-PG1. Since this function has already been discussed adequately in [7], it will not be repeated here.

Of greater interest with regard to the error-compensation procedure is the function of MINI2 in block 18 of the flowchart of PROCCE. In that block, an entry point, INITGS, of MINI2 is called which performs an initialization of the CSPFE vectors of the network. Although this aspect of the subroutine MINI2 was mentioned in [7], it was not discussed in detail. The rest of this section will explain the results of calling INITGS.

During the main body of the error-compensation procedure (i.e., during the call to RCEC in block 19 of the flowchart of PROCCE) the compatible sets of permissible functions are determined by assigning logical *'s, 0's, 1's, $\underline{0}$'s, and $\underline{1}$'s to elements of GSMALL (recall that the array GSMALL is used to store the CSPFE vectors). There is usually considerable freedom in making these assignments, and consequently a large variety of collections of compatible sets for the entire network can be produced. Despite this freedom though, there are certain assignments of logical 1's and 0's to certain entries of GSMALL which are predetermined by the configuration of the network and by the algorithm to be applied. These necessary assignments are made during the initialization step in order to avoid making certain unnecessary assignments (to particular elements of GSMALL) later, as would otherwise occur.

It is important, however, to remark that the existence of such predetermined entries is only possible under the assumption that only a *single* call to RCEC will follow the initialization. Partly for this reason, INITGS is always called to re-initialize the CSPFE vectors immediately before each call to RCEC (see blocks 18 and 19 of the flowchart of PROCCE). Two or more successive calls to RCEC (without changing the initialization assignments) might change the network configuration so that any such initialization would be invalid. In any event, a call to RCEC usually alters the network configuration so that all of the CSPFE vector entries calculated during that call by RCEC itself must be recalculated (in the next call to RCEC), so some sort of initialization must be performed before each call anyway.

As previously mentioned, five different logical entries, *, 0, 1, $\underline{0}$, and $\underline{1}$, may appear as components of the different CSPFE vectors. The precise meanings of these five logical entries are discussed in [5], and they will also become clear during the discussion of the error-compensation subroutine (Section 2.3). In general terms, however, the meanings are as follows:

* - indicative of a "don't-care" component. In other words, every permissible function of the corresponding CSPFE may

have either a 0 or a 1 for this component.

0 - indicative of a required 0 component. Every permissible function of the corresponding CSPFE must have a 0 for this component position.

1 - indicative of a required 1 component. Every permissible function of the corresponding CSPFE must have a 1 for this component position.

$\underline{0}$ - indicative of currently required 0 that should preferably be a required 1 instead (this is called a "0-error"). Although every permissible function of the corresponding CSPFE currently has a 0 for this component, it would be desirable to change the network configuration so that a 1 could be demanded instead.

$\underline{1}$ - indicative of currently required 1 that should preferably be a required 0 instead (this is called a "1-error"). Although every permissible function of the corresponding CSPFE currently has a 1 for this component, it would be desirable to change the network configuration so that a 0 would be demanded instead.

To understand the coding of these logical values into the actual values used by the program, the concept of "covering" must be understood. Due to the nature of NOR gates, a 1 appearing on any of the input lines to a gate will cause a 0 output of that gate. Such a 1 is called a *cover* of that 0, and the 0 is said to be *covered* by that 1. Although a 0 output may be covered by several 1's (appearing on different input lines), only a single cover is actually required to guarantee the 0 output. For a given gate, certain 0 components of its CSPFE vector (assuming that its CSPFE vector has already been calculated somehow - at least partially), representing required 0 outputs of the gate, each have only a single 1 cover. In each such case, the single 1 covering the 0 is called an *essential 1*. These essential 1's form the basis of the initialization steps performed by calling INITGS.

The correspondence between the logical values *, 0, 1, $\underline{0}$, and $\underline{1}$ and the actual (coded) values used in the program is shown in Table 2.2.1.1 (the symbols α and β used in the table represent variable integer values which may appear as the computer's internal representation of the logical value 0 of a CSPFE component).

LOGICAL VALUE OF CSPFE COMPONENT		ACTUAL VALUE OF CSPFE COMPONENT
*	→	0
0	→	α, where $0 > \alpha > -1000$ $\alpha = -100$ means this component has an unknown number of covers $\alpha = -200$ means this component has 2 or more covers. $\alpha = -\beta \neq -100, -200$ means this component has an essential 1 cover from gate or external variable β
1	→	1
$\underline{0}$	→	-1100
$\underline{1}$	→	1001

Table 2.2.1.1 Coding of logical values for components of CSPFE vectors.

The sequence of events when INITGS is called is shown in Figure 2.2.1.1. INITGS assumes certain preliminaries have already been completed before it is called. For example, GORDER is assumed to be updated and

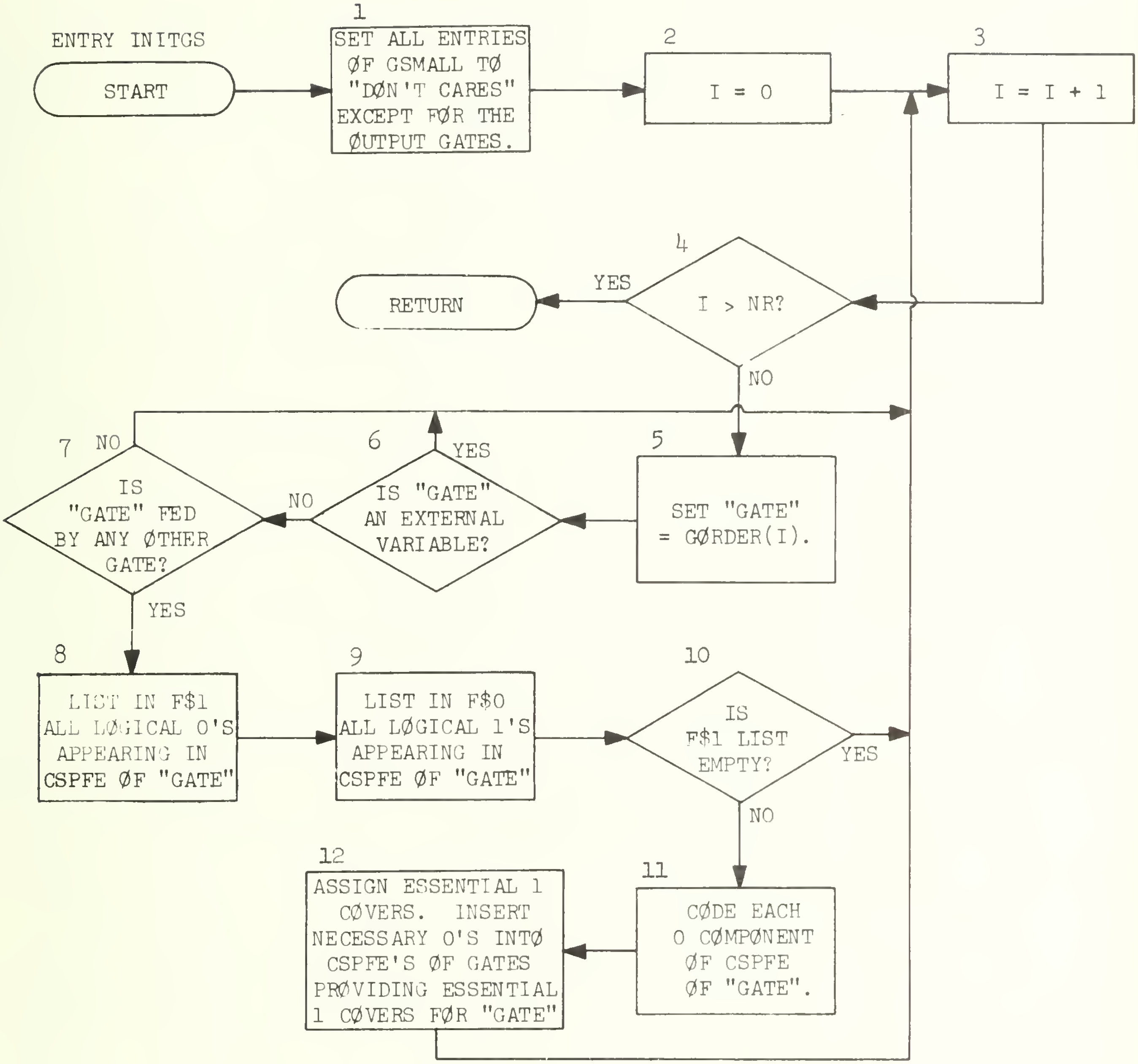

ENTRY INITGS
START
1
SET ALL ENTRIES ØF GSMALL TØ "DØN'T CARES" EXCEPT FØR THE ØUTPUT GATES.
2
I = 0
3
I = I + 1
4
I > NR?
YES
RETURN
NO
5
SET "GATE" = GØRDER(I).
6
IS "GATE" AN EXTERNAL VARIABLE?
YES
NO
7
IS "GATE" FED BY ANY ØTHER GATE?
NO
YES
8
LIST IN F$1 ALL LØGICAL 0'S APPEARING IN CSPFE ØF "GATE"
9
LIST IN F$0 ALL LØGICAL 1'S APPEARING IN CSPFE ØF "GATE"
10
IS F$1 LIST EMPTY?
YES
NO
11
CØDE EACH 0 CØMPØNENT ØF CSPFE ØF "GATE".
12
ASSIGN ESSENTIAL 1 CØVERS. INSERT NECESSARY 0'S INTØ CSPFE'S ØF GATES PRØVIDING ESSENTIAL 1 CØVERS FØR "GATE"

Figure 2.2.1.1 Flowchart of a section of the subroutine MINI2 (beginning at entry point INITGS).

GSMALL is assumed to already contain the correct values for the CSPFE vectors of the output gates (this is done by PROCCE).

In block 1 of the flowchart in Figure 2.2.1.1, all of the entries in GSMALL, except for those corresponding to the network output gates (since these have already been determined), are set to "don't cares" in preparation for the initialization.

Block 2 sets a counter, I, to the value 0. Thereafter, each pass through block 3 will increase I by 1. If I should exceed NR (the total number of gates and external variables), as tested in block 4, a return is made to PROCCE.

Otherwise, block 5 sets the variable GATE equal to GORDER(I). GATE is the current gate under consideration. Elements of the CSPFE vectors of the gates which feed GATE will be initialized according to the already initialized CSPFE vector of GATE itself (the ordering contained in the array GORDER guarantees that the CSPFE vector of GATE will be completely initialized before the CSPFE vectors of any of its predecessors).

Blocks 6 and 7 test whether or not GATE is an external variable or an isolated gate. If it is either, the program returns to block 3 to increase I and select a new GATE.

A list is made in block 8 of the component positions of all the logical 0's currently appearing in the CSPFE of GATE. These are stored in the 1-dimensional array F$1.

Similarly, block 9 lists the positions of all of the logical 1's and stores them in the array F$0.

If F$1 contains an empty list, GATE has no logical 0's in its CSPFE to be covered by its predecessors, and the covering step in block 12 can be skipped. The test is made in block 10.

By checking the number of predecessors which cover each logical 0 in GATE's CSPFE vector, block 11 determines the actual coding for each of these logical 0's individually (according to Table 2.2.1.1).

This coding is used by block 12 which assigns values to (i.e., initializes) certain components of the CSPFE vectors of GATE's predecessors. Each logical 0 in GATE's CSPFE vector whose actual coding is of the $-\beta$ ($\beta \neq 100, 200$) type has an essential 1 cover. In other words, each logical 0 of this type is covered by the function from only a single gate.

Making the assumption that every logical 0 or 1 appearing in the CSPFE vector of GATE can be shown to be a required value assignment regardless of the finally chosen sets of CSPFE vectors (i.e., for any choice of compatible sets of permissible functions for the gates of the network, these components of the CSPFE vector of GATE will always have the same assigned value), one can assert that the essential 1 covers of those logical 0's are also required regardless of the finally chosen CSPFE vectors (assuming no change in the network configuration). Thus justified, an assignment of a logical 1 is made to the CSPFE vector of a predecessor, PRED, of GATE in a component position corresponding to an essential 1 cover provided by PRED for a 0 in GATE's CSPFE vector. This is done by block 12 for every 0 in the CSPFE vector of GATE.

Furthermore, since every predecessor, PRED, of GATE which provides an essential 1 cover for at least one 0 of GATE's CSPFE vector cannot be disconnected from GATE without introducing a new error into the network, it is clear that the CSPFE vector of PRED must also contain a logical 0 in every component position where a logical 1 appears in GATE's CSPFE. These assignments are also made in block 12.

Generally the initialization of GSMALL (i.e., the CSPFE vectors) can be seen as a propagation of required values of certain CSPFE vector components from the output gates through the gates feeding them, through the gates feeding these gates, etc.: The complete CSPFE vectors for the network output gates are known completely from the beginning. For the gates providing essential 1 covers for certain 0's in the CSPFE's of the output gates, one can make assignments of 0's and 1's to their own CSPFE vectors which are valid independent of whatever algorithm is used later (after the initialization step) to make a complete assignment to all components of all CSPFE vectors. This step is then repeated for the gates providing essential 1 covers for the gates providing essential 1 covers for the output gates, and so on.

2.2.2 Potential output table (POT)

In order to compensate for error-components at the CSPFE of a gate, functions currently realized at other gates and external variables may be used according to the basic transduction procedure using CSPFE's discussed in [5]. In order to make the error-compensation more flexible, the concepts of potential outputs and potential output tables (POT) were introduced in [5], and a procedure utilizing a potential output table was also given. This section will briefly explain these concepts and discuss in some detail the implementation of the potential output table in programs NETTRA-E1, -E2, and -E3. (For details of these concepts see [5]).

A potential output from a gate GI is a function realized at GI by connecting additional inputs to GI. This potential output of GI differs from the function currently realized at GI, and therefore can be used to compensate for errors in a certain gate to which the current output function at GI is not connectable (i.e., connecting GI to that gate will change some non-error components in its CSPFE). There may be two major problems in using potential outputs. One is the problem of over-compensation caused by using too sophisticated potential outputs (i.e., too many modifications are required in producing potential outputs) that the number of error components in other gates may increase and it becomes too difficult to completely compensate for them. This problem can be avoided by restricting the types of modifications used in producing potential outputs. As explained in [5], only potential outputs which can be realized by adding connections satisfying triangular conditions are allowed in the error compensation procedure. Another potential problem is the additional computational complexity required by searching for potential outputs. This problem can be serious if a complete searching of potential outputs is required each time when a potential output is desired. This problem of time-consuming search cannot be completely avoided (if potential outputs are to be used) but can be reduced to a certain degree by employing a potential output table (POT). The POT is a list of selected potential outputs and the information on how to construct each potential output. For the sake of convenience, functions which are currently realized at input terminals (external variables) and gates are also listed. In the beginning of the error compensation process, all potential outputs satisfying certain triangular conditions are searched and listed in the form of a table. During the error-compensation process for a particular gate GI, the candidates for new inputs to GI (strongly effectively E-connectable functions with respect to the CSPFE of GI) are selected from POT. Therefore, at the expense of memory space (for storing POT), the relatively time consuming process of searching for potential outputs can be replaced by a simple table look-up process.

The potential output table used in NETTRA-E1, -E2, and -E3 is organized as follows.

(1) The potential output table (POT) is stored in a two dimensional array POTAB(200,42). The first argument of POTAB is the entry number. For example, the information on the PTR-th entry of POT is stored in POTAB(PTR, 1) ~ POTAB(PTR, 42).

(2) Each entry, corresponding to one potential output, contains the following information:

(a) POTAB(PTR, 1) ~ POTAB(PTR, 32): the actual function (vector) of the PTR-th potential output in a truth table form.

(b) POTAB(PTR, $GT): the gate label of gate GI at which the PTR-th potential output is realized ($GT is an integer constant and has the value 33).

(c) POTAB(PTR, $LTH): the number of connections to be added to gate GI in order to realize the PTR-th potential output ($LTH is an integer constant and has the value 34). POTAB(PTR, $LTH) may have the value 0 through 6.

(d) POTAB(PTR, $LTH+1) ~ POTAB(PTR, $LTH+6): the list of external variables and/or gates whose outputs must be connected to GI in order to realize the PTR-th potential output.

(e) POTAB(PTR, $PW) and POTAB(PTR, $NOE): the preference weight and the number of one errors, in the PTR-th potential output, respectively. Since these two values are defined with respect to the CSPFE of a particular gate, these two fields are used only during the error-compensation process when a gate has been selected. ($PW and $NOE are integer constants and have the values 41 and 42, respectively).

(3) Entries of POT are divided into several blocks of consecutive entries. Each block contains the potential outputs realized at one particular gate. In other words, every entry in the same block has the same POTAB(PTR, $GT), and for every gate or external variable in the network there is a corresponding block containing the potential outputs realized at that gate or external variable (a block may consist of only one entry).

(4) For each gate or external variable in the network, there are two pointers PPOTAB and LPOTAB both of which are one dimensional arrays. PPOTAB(GI) indicates the starting entry of the block corresponding to GI, whereas LPOTAB indicates the last entry of the same block.

(5) In the block corresponding to gate or external variable GI, the entries are listed in the following order:

(a) The function currently realized at GI, and therefore POTAB(PPOTAB(GI), $LTH) = 0

(b) The functions realized at GI with one additional connection (i.e., POTAB(PTR, $LTH) = 1). For the sake of convenience, these entries are called simple entries.

(c) Entries realized at GI with POTAB(PTR, $LTH) $\geq$ 2.

An entry with POTAB(PTR, $LTH) $\geq$ 2 is called a composite entry because it can be obtained by an operation with two other entries with the same POTAB(PTR, \$GT). For example, if the PTRA-th and PTRB-th entries satisfy POTAB(PTRA, \$GT) = POTAB(PTRB, \$GT) = GI, a composite entry PTR can be generated as follows.

(1) POTAB(PTR, I) = POTAB(PTRA, I) ∧ POTAB(PTRB, I), for I = 1, 2, ..., 32.

(2) POTAB(PTR, \$GT) = GI.

(3) POTAB(PTR, \$LTH) = POTAB(PTRA, \$LTH) + POTAB(PTRB + \$LTH).

(4) POTAB(PTR, \$LTH + I) = POTAB(PTRA, \$LTH + I), for I = 1, ..., POTAB(PTRA, \$LTH).

(5) POTAB(PTR, \$LTH + I) = POTAB(PTRB, \$LTH + I - POTAB(PTRA, \$LTH)), for I = POTAB(PTRA, \$LTH) + 1 , ..., POTAB(PTRA, \$LTH) + POTAB(PTRB, \$LTH).

In programs NETTRA-E1, -E2, and -E3, the potential output table for a given network is generated by subroutine POT which realizes the following procedure.

Procedure for Generating Potential Output Table

Step 1. Initialize

Set PPOTAB(GI) = 0 for GI = 1, ..., N + R, where N and R are the numbers of external variables and gates in the network, respectively. Set POINTR = 1 (POINTR is a pointer indicating the next entry to be generated).

Step 2. Selection of gates

According to the level assigned to each gate and external variable (i.e., a gate in a higher level precedes a gate in a lower level), select a gate or an external variable, GI, in the network. If all gates and external variables have been considered, the procedure terminates.

Step 3. Set the first entry for GI

Set PPOTAB(GI) = POINTR.

Copy the present function realized at GI into POTAB(POINTR, 1) ~ POTAB(POINTR, N2), where N2 is the number of input combinations (if completely specified, N2 = 2**N).

Set the following values:

POTAB(POINTR, $GT) = GI,

POTAB(POINTR, $LTH) = 0, and

POINTR = POINTR + 1.

If GI ≤ N, go to Step 6.

Step 4. Find the simple entries realized at GI

Step 4-1 Select a gate or an external variable, GJ, whose level number is not lower than that of GI. If all such GJ's have been considered, go to Step 5.

Step 4-2 Check whether or not GJ is connected to all immediate successors of GI. If not, go to Step 4-1.

Step 4-3 Check whether or not connecting GJ to GI produces a function which differs from the function of GI and from the functions produced by connecting previous GJ's to GI. If not, go to Step 4-1.

Step 4-4 Make the POINTR-th entry as follows..

POTAB(POINTR, $GT) = GI

POTAB(POINTR, $LTH) = 1

POTAB(POINTR, $LTH + 1) = GJ

POINTR = POINTR + 1

Go to Step 4-1.

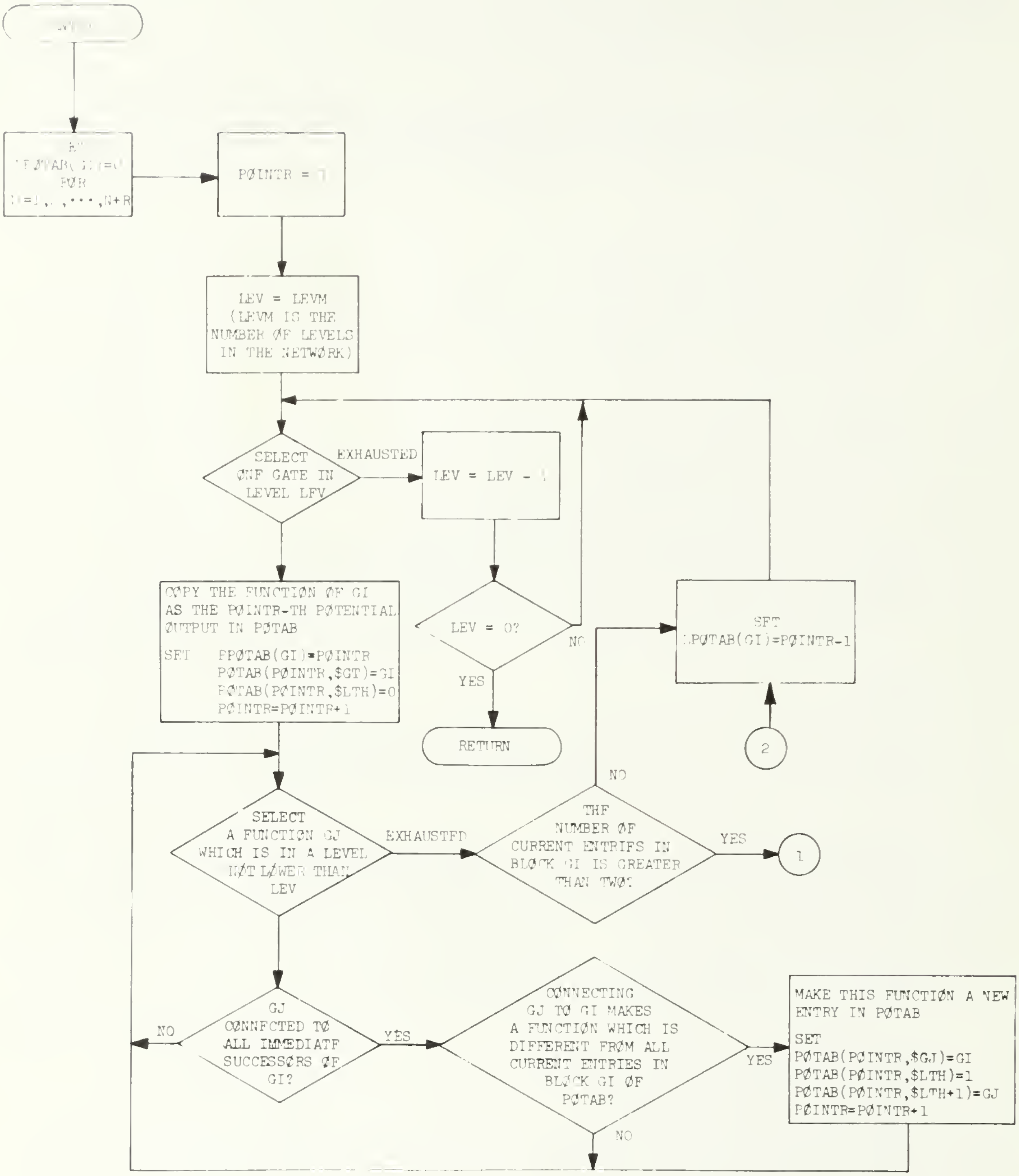


Figure 2.2.2.1 Flowchart of subroutine POT

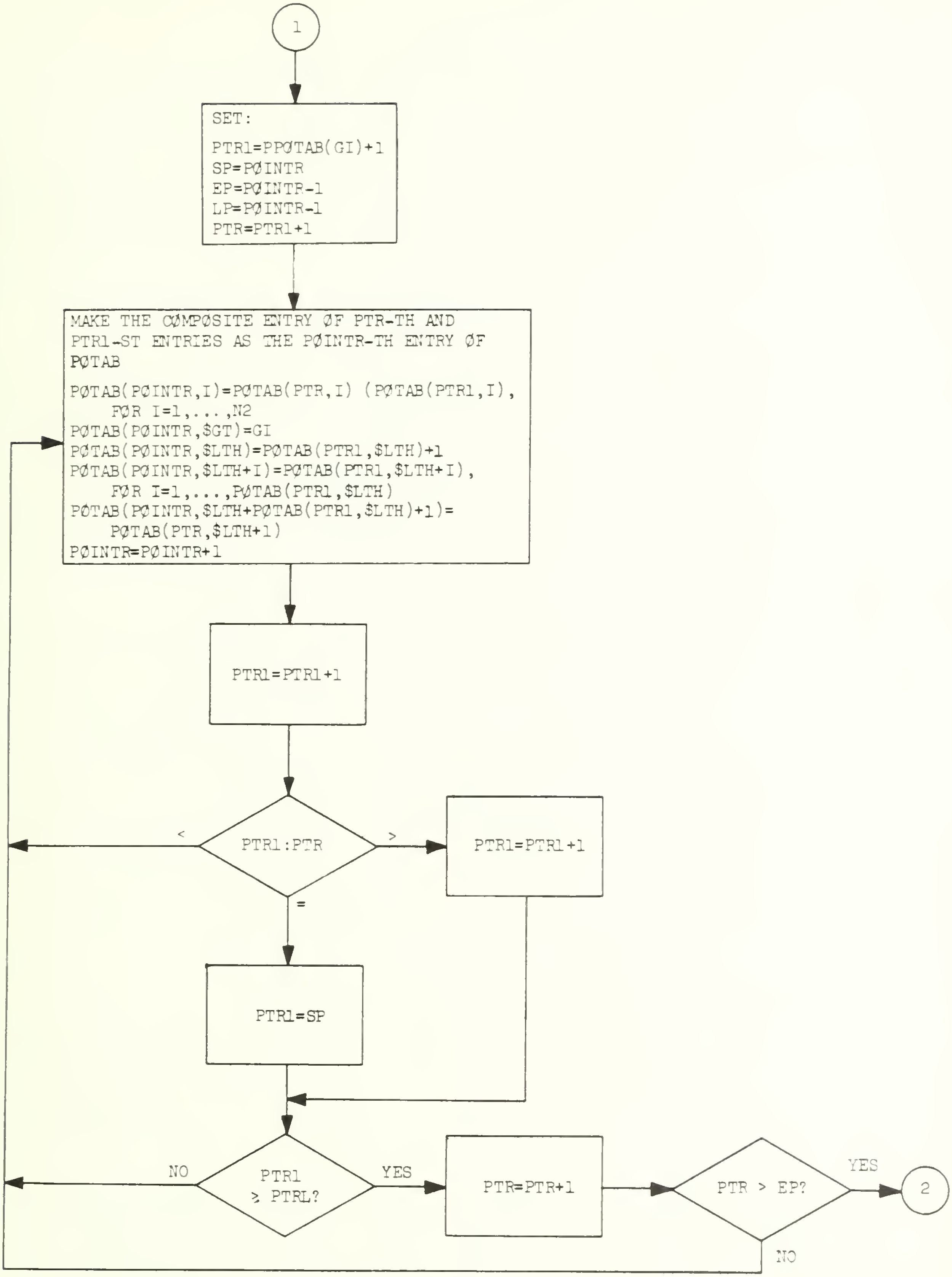

1
SET:
PTR1=PPØTAB(GI)+1
SP=PØINTR
EP=PØINTR-1
LP=PØINTR-1
PTR=PTR1+1
MAKE THE CØMPØSITE ENTRY ØF PTR-TH AND
PTR1-ST ENTRIES AS THE PØINTR-TH ENTRY ØF
PØTAB
PØTAB(PØINTR,I)=PØTAB(PTR,I) (PØTAB(PTR1,I),
FØR I=1,...,N2
PØTAB(PØINTR,$GT)=GI
PØTAB(PØINTR,$LTH)=PØTAB(PTR1,$LTH)+1
PØTAB(PØINTR,$LTH+I)=PØTAB(PTR1,$LTH+I),
FØR I=1,...,PØTAB(PTR1,$LTH)
PØTAB(PØINTR,$LTH+PØTAB(PTR1,$LTH)+1)=
PØTAB(PTR,$LTH+1)
PØINTR=PØINTR+1
PTR1=PTR1+1
<
PTR1:PTR
>
PTR1=PTR1+1
=
PTR1=SP
NO
PTR1
≥ PTRL?
YES
PTR=PTR+1
PTR > EP?
YES
2
NO

Figure 2.2.2.1 Continued.

Step 5. Generate composite entries to be realized at GI

If there is no more than one simple entry for GI, go to Step 6; otherwise generate composite entries as follows. (S denotes the total number of simple entries, which are the entries PPOTAB(GI) + 1 through PPOTAB(GI) + S in the potential output table, and POINTR always points at the next entry to be entered).

Step 5-1 Set I = 2 and E = S.

Step 5-2 Generate a composite entry from the (PPOTAB(GI) + I)-th and the (PPOTAB(GI) + J)-th entry, for J = 1, ..., I - 1.

Step 5-3 Generate a composite entry from the (PPOTAB(GI) + I)-th entry and the (PPOTAB(GI) + J)-th entry, for J = S + 1, ..., E(if E ≤ S, this step is skipped).

Step 5-4 Set I = I + 1. If I > S, go to Step 6; otherwise set E = POINTR and go to Step 5-2.

Step 6. Set the pointer indicating the last entry in block GI

Set LPOTAB(GI) = POINTR - 1, and go to Step 2.

The flowchart of this procedure is shown in Figure 2.2.2.1.

2.3 Error-Compensation Subroutines

Error compensation procedure is coded into a FORTRAN subroutine named RCEC which stands for Replacement of Connections for Error-Compensation and five supporting subroutines: CALS1, RPLCF, FORC, ORDRQ2, and CONECT. RCEC is the central part of the transduction programs NETTRA-E1, -E2, and -E3. This subroutine needs, as parameters, a NOR network which does not realize the desired output functions along with the initialized CSPFE's for each gate (explained in Section 2.2.1). When it is entered, it selects one gate at a time whose CSPFE has been completely calculated and tries to compensate for error-components in the CSPFE of the selected gate. As a result of the error-compensation for this particular gate, if some error-components are compensated or output functions of some gates are changed, this subroutine will return to PROCCE, the calling subroutine in NETTRA-E1, -E2, and -E3. PROCCE will then recalculate the output functions of the network to check whether or not this new network realizes the desired functions. If it does, the original network has already been transduced to a desired new network; otherwise PROCCE will recalculate the potential output table and call subroutine RCEC again to apply the error-compensation procedure to this slightly modified network. On the other hand, if no error-components in the CSPFE of the selected gate can be compensated, the CSPFE of the selected gate will be propagated to its inputs. If all gates have been considered in this manner, it means no error-components can be compensated in this application of RCEC (otherwise it would have returned to PROCCE). In this case, the subroutine returns to PROCCE unsuccessfully.

The error-compensation procedure consists of several subprocedures. These subprocedures and two supporting subroutines will be discussed in some detail in Sections 2.3.1 and 2.3.2, respectively. Section 2.3.3 will be devoted to the discussion of the propagation of CSPFE's, and Section 2.3.4 will summarize the entire procedure and present the flowchart.

2.3.1 Compensation of error-components for a particular gate

The error-compensation procedure considers only one gate at a time. When a gate is selected, its CSPFE must be completely calculated. Therefore, at the beginning of the procedure, only first level output gates can be selected. As the calculation goes on, a gate becomes selectable only when all its immediate successors have been selected. Thus the ordering of selection can be made according to the gate level assigned to each gate. The ordering also could take into consideration the number, type and degree of the error-components in the CSPFE of each gate when more than one gate is selectable as explained in [5].

After a gate GI has been selected, the procedure concentrates on compensating for error-components in the CSPFE of gate GI by (1) removing redundant input connections, (2) substituting for input connections, and (3) adding connections to compensate for 1-error-components. The first two types of operations are aimed at compensating for 0-error-components whereas the third one 1-error-components. In all cases, the number of error-components in the CSPFE of gate GI will never increase after applying these operations. In addition to the number of error-components, the degree of an error-component is also an important criterion in deciding which connections are to be added or disconnected. The degree of an error-component is defined only for 0-error-components to indicate how difficult this 0-error-component is to be compensated. If a 0-error-component in the CSPFE of gate GI is covered by only one input connection of the gate, this error is considered easier to be compensated since the proper substitution for this input connection or the compensation for the corresponding 1-error-component in that input can compensate for this 0-error-component. This type of 0-error-components are called primary 0-error-components[†]. On the other

[†] The Definition 4.3.2.3 of [5] defines primary 1-error-components which are the ones which cover primary 0-error-components defined here. For the convenience of discussion both are regarded as primary error-components in this paper.

hand, if a 0-error-component is covered by more than one input , it is considered more difficult to be compensated either at this stage when the gate having this 0-error-component is under consideration or later when the immediate predecessors covering this 0-error-component are under consideration. This type of 0-error-components are called secondary error components.

Among the three types of operations mentioned earlier, the last two require the connection of functions to the selected gate GI. The set of candidates for those functions must be strongly effectively E-connectable functions with respect to the CSPFE of gate GI. Among these candidates the functions which have been actually selected to be connected to GI as inputs must also satisfy the prohibition conditions (Lemma 3.2 of [5]) with each another. These conditions are examined each time when a function has been selected, and the functions which do not satisfy one of these conditions with this selected function will be prohibited from being selected.

The above conditions are required for the functions to be connected to gate GI both in the second (substitution of connections) or the third type (addition of connections) operation. In addition to these conditions, the replacing functions and the replaced functions must satisfy the following condition. That is, the addition of the entire set of replacing functions will make the set of to-be-replaced functions E-disconnectable with respect to the CSPFE of gate GI. This condition guarantees that the number of 0-error-components never increases. In the actual procedure, the operation of the substitution for input connections consists of three subprocedures. Each subprocedure is aimed at compensating for a particular type of error-components. Additional conditions are required in each subprocedure in order to actually compensate for some error-components, and will be discussed later in this section. For the third type of operation, each of the added connections must have at least one 1-component which covers a 1-error-component in the CSPFE of gate GI since this operation is aimed at compensating for 1-error components.

The input connection substitution problem is essentially a covering problem. All 0-components in CSPFE of gate GI must still be covered after the substitution, and the number of 0-error-components covered by the new input set (the replacing functions and the remaining functions) is to be minimized. Although the optimal solutions can be obtained by solving this covering problem, the time required for deriving optimal solutions would usually be too excessive, especially when the number of candidates for substituting functions is very large. Furthermore, the local optimization, i.e., the optimization concerning only the selected gate GI is not necessarily the optimization with respect to the entire error compensation procedure. Based on these considerations, a heuristic method for substitution subprocedures is used.

The error compensation procedure contains the following six subprocedures, among which, subprocedures (2), (3), and (4), are substitution subprocedures.

(1) Remove redundant connections.

The redundant connections are the input connections which are E-disconnectable with respect to the CSPFE of the selected gate GI. Since it has no essential 1-components to cover 0-components in the CSPFE of GI, the removal of this connection will not increase the number of error-components.

The number of 0-error-components or the degrees of some 0-error-components in the CSPFE of the selected gate may decrease resulting from the removal of redundant connections. It should be noted that the redundant connections are removed before substitution is considered but the removed input connection may be reconnected in the substitution subprocedures.

(2) Substitution for input connections from external variable with error-components.

If an input connection from an external variable has a 1-component which covers a 0-error-component in the CSPFE of gate GI, this 0-error-component can never be compensated unless this input connection to gate GI is removed. Therefore, this type of input connections must be considered for substitution prior to other input connections. Since the main purpose of this subprocedure is to replace the input connections having uncompensatable errors, the strongly effectively E-connectable functions (for simplicity, strongly effectively E-connectable functions will henceforth be referred to as connectable functions) without uncompensatable error-components, i.e., the connectable function from external variables without error-components and connectable function from gates with or without error-components may be used as the candidates for the substituting functions.

(3) Substitution of input connections from gates with primary errors.

If a 0-error-component in the CSPFE of the selected gate is covered by only one input function, it is called a primary error as it may be corrected by substituting for that input function only. In this case, the connectable functions which have a 0-component corresponding to the primary 0-error-component under compensation are candidates for the substitution. Once a primary error is eliminated by substitution, the candidates for later substitutions should be limited to those functions which have 0-components corresponding to the compensated primary 0-error-component.

(4) Substitution for input connections by functions without error-components.

This subprocedure substitutes connectable functions which have no 1-components corresponding to 0-error-components in the CSPFE of the selected gate GI for functions which have at least one 1-component corresponding to a 0-error-component in the CSPFE of the gate GI. As a result of the substitution the degrees of some 0-error-components may be reduced and this may make the total number of error-components in the input functions smaller. If the degree of an error is reduced to zero, this error of the selected gate has been compensated.

(5) Adding connections to compensate for 1-error-components.

A 1-error-component in the CSPFE of the selected gate GI is easy to be compensated if there is a connectable function whose corresponding component is a 1. Some of the 1-error-components may have been already compensated during the compensation for 0-error-components by substitutions. Since earlier subprocedures are aimed at compensating for 0-error-components, however, there are still possibilities to compensate for 1-error-components especially when there is no 0-error-component in the CSPFE of gate GI. In this subprocedure, the candidates are those functions which have a 0-component corresponding to each 1-component (including the corrected 0-error-components if any) in the CSPFE of gate GI.

(6) Adding redundant input connections from external variables.

This subprocedure does not belong to any of the three types of operations for error-compensation since it is not aimed at compensating for error-components in the CSPFE of the selected gate GI, but rather, at loosening the requirements of the predecessors of gate GI to make error-compensation at later steps easier.

In subprocedure (2), the inputs from external variables with error-components have been considered for replacement in the first place since an error-component in an external variable can never be compensated unless it is removed. A redundant input from an external variable without error-components, however, will help to loosen the requirements for the predecessors of the selected gate GI if this input from external variable has some 1-components corresponding to the 0-components in the CSPFE of gate GI. Therefore, this type of redundant external variables (i.e., those without anticipated error-components) should be connected to gate GI. In addition, the external variables with some anticipated 1-error-components which cover the corresponding 0-error-components are also added if these 0-error-components are not primary errors (i.e., the addition of these redundant inputs from external variables will not decrease the number of primary 0-error-components in the CSPFE of the selected gate). This seems to contradict the objective of subprocedure (2), i.e., removing external variables with error-components by any means, but it should be noted that if some error-components can be compensated later, this gate will be considered again when subroutine RCEC is reentered. At that time, the redundant external variables, if still redundant, will be removed by subprocedure (1), and may not be added again if it would cover primary 0-error-components.

These subprocedures will be explained in some detail in Section 2.3.3.

2.3.2 Supporting subroutines for substitution

In the error-compensation procedure, there are three subprocedures which substitute a subset of candidates for a subset of input functions of a selected gate GI. Since these subprocedures are similar in nature, they are coded to share two subroutines, CALS1 and RPLCF. Subroutine CALS1 calculates the to-be-replaced subset of certain input functions when a set of candidates are given. Let the set of input functions to be removed be denoted with S, and the set of candidates for the functions to be added with S2†. The subroutine CALS1 will calculate a subset S1 of S which can be replaced by S2. For example, in subprocedure (2) the set S contains all external variable inputs which cover some 0-error-components, and the set S2 contains all connectable functions except the connectable external variables which have 1-components corresponding to some 0-error-components in the CSPFE of the selected gate GI. The subroutine CALS1 checks whether or not every essential 1-component in each function of S can be covered by S2. If all essential ones in a function of S can be covered by set S2, this function is replaceable, and therefore should be placed in set S1. Along with set S1, the components which must be covered by one of the substituting functions are stored in set T1, which will be referred to by subroutine RPLCF.

† In reference [5], symbols S, S_1, S_2, and S_3 were not used in order to avoid the possible confusion with $S(v_i)$, the set of successors for input terminal or gate v_i. Instead of S, S_1, S_2, and S_3, symbols Q, Q_1, Q_2, Q_3 were used to indicate a set of input functions containing the candidates to be replaced, a subset of Q which can be actually replaced, a set of candidates replacing Q_1, and a subset of Q_2 which can actually replace Q_1, respectively. For better correspondence with the notation in the actual programs NETTRA-E1, NETTRA-E2, and NETTRA-E3, the symbols S, S1, S2 and S3 are used in this paper in place of Q, Q_1, Q_2, and Q_3, respectively.

The procedure of subroutine CALS1 is as follows.

Step 1. Selection of functions

Take a function GP from set S. If all functions in S have been considered,return to the calling procedure (subroutine RCEC in NETTRA-E1, -E2, or -E3).

Step 2. Check replaceability

For each 0-component in the CSPFE of GI, the gate under consideration, which is covered only by input function GP, check whether or not set S2 covers it. If not, GP is not replaceable and go to Step 1.

Step 3. List essential ones

Add the positions of all essential 1-components in the function of GP into set T1, which is the list of the positions to be covered by the substituting functions.

Step 4. Update

Add GP into set S1. Remove GP from the list of current input functions of GI. Remove GP from set S. Go to Step 1.

The flowchart of subroutine CALS1 is shown in Figure 2.3.2.1.

Subroutine RPLCF is called immediately after CALS1 has returned. RPLCF selects a subset S3 of S2 which is needed to replace functions in set S1. Therefore, sets S3 and S1 must satisfy the conditions for substitutions described in Section 2.3.1. Since the selection of S3 is essentially a covering problem, as mentioned previously, a heuristic procedure is used. The candidates for the replacement of S1 (i.e., set S2) are stored according to the ordering based primarily on the number of

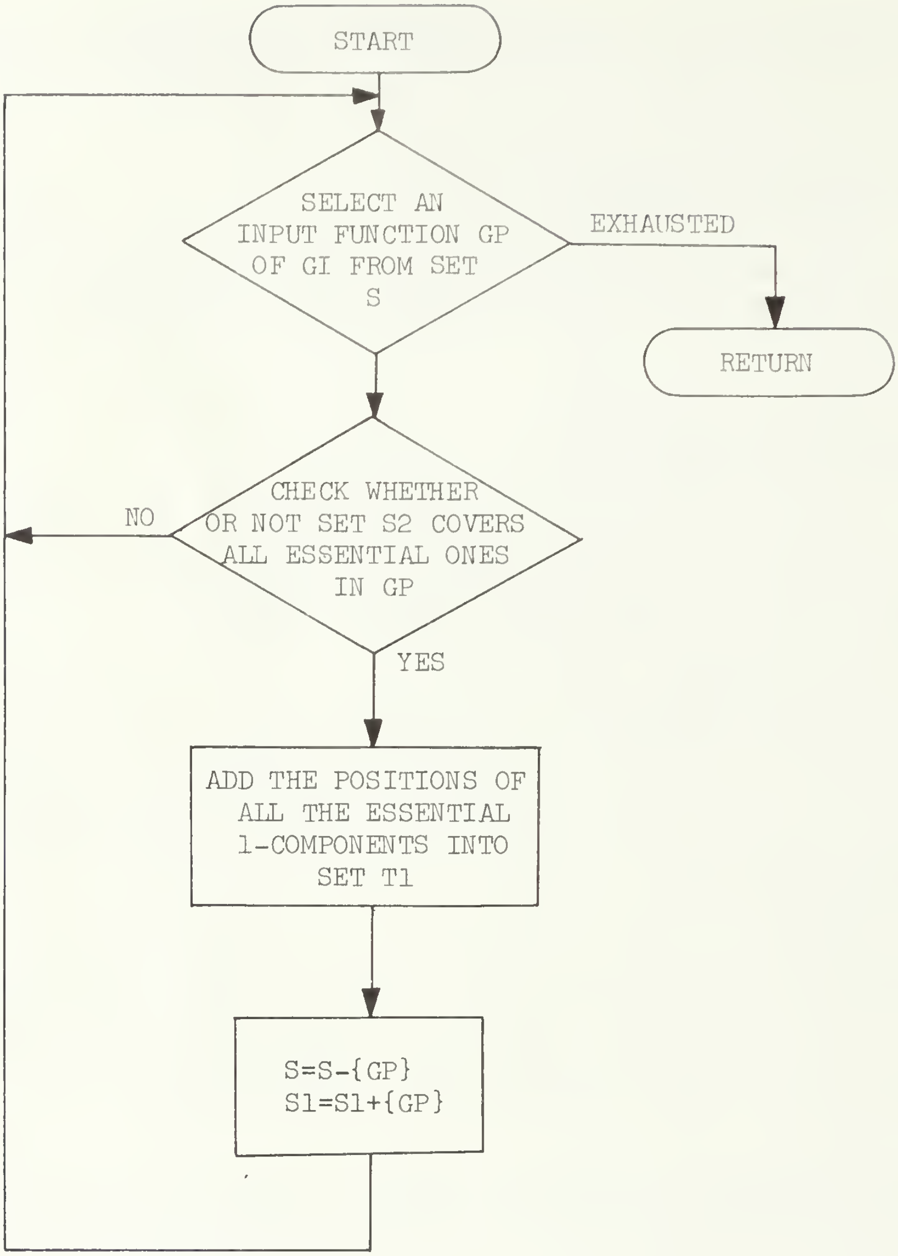

START
SELECT AN INPUT FUNCTION GP OF GI FROM SET S
EXHAUSTED
RETURN
CHECK WHETHER OR NOT SET S2 COVERS ALL ESSENTIAL ONES IN GP
NO
YES
ADD THE POSITIONS OF ALL THE ESSENTIAL 1-COMPONENTS INTO SET T1
S=S-{GP}
S1=S1+{GP}

Figure 2.3.2.1 Flowchart of subroutine CALS1.

anticipated error-components (if connected) and secondarily the number of 1-components covering 0-components or 1-error-components. The procedure examines each candidate according to this ordering to check if it covers any components in T1 which are not covered by any selected candidates. If it does, the candidate will be added into set S3 which is the set of functions substituting for inputs in set S1. As mentioned in Sections 2.2.2 and 2.3.1, the candidates in set S2 may not be compatible with one another, so each time when a candidate is added into set S3, all other candidates which are not compatible with it are prohibited from being selected. Because of the prohibition, the remaining functions in set S2 and functions in set S3 may no longer cover set T1. In this case, the functions prohibited by the latest selection are permanently prohibited, and the set S2 is reconstructed. The procedure then calls subroutine CALS1 (from subroutine RPLCF) and reenter RPLCF itself to repeat this procedure of substitution. As a final result, a subset of input functions may be replaced by a subset of candidates in set S2, and the conditions listed in Section 2.3.1 are satisfied.

The procedure of subroutine RPLCF is as follows.

Step 1. Initialize

Set S3 = $\emptyset$.

Step 2. Selection of a candidate

Select a function, PTR, from set S2 which is the PTR-th entry of POT and is realized at GT.

Step 3. Check usefulness

Check whether or not this function covers any components in set T1 which are not covered by any functions already in set S3. If not, go to

Step 2.

Step 4. Prohibition

Prohibit functions which are realized either at gate GT or at some other gate GX requiring the addition of the connection from GT to GX.

Step 5. Update

Temporarily add this function into set S3. Remove components covered by this function from set T1. If T1 = ∅, go to Step 8.

Step 6. Check replaceability

Check whether or not remaining functions in set S2 still cover all remaining components in set T1. If yes, set S3 = S3 $\cup$ {PTR} and go to Step 2.

Step 7. Recalculate set S1

Restore set S2 but delete from S2 the functions just prohibited in Step 4. Call CALS1 to recalculate S1 and associated T1. Go to Step 1.

Step 8. Substitution

Connect functions in S3 to gate GI. Disconnect the input functions in set S1 from gate GI. Return to the calling procedure (subroutine RCEC in NETTRA-E1, -E2, and -E3).

The flowchart of this subroutine is shown in Figure 2.3.2.2.

Beside the two subroutines mentioned above, there are three other supporting subroutines: FØRC, ØRDRQ2, and CONECT.

FORC has an argument GJ. When it is called,the connection from GJ to GI which is the gate under consideration is examined. If the connection from GJ to GI is E-disconnectable with respect to the CSPFE of GI, it will be removed from the network; otherwise it will do nothing.

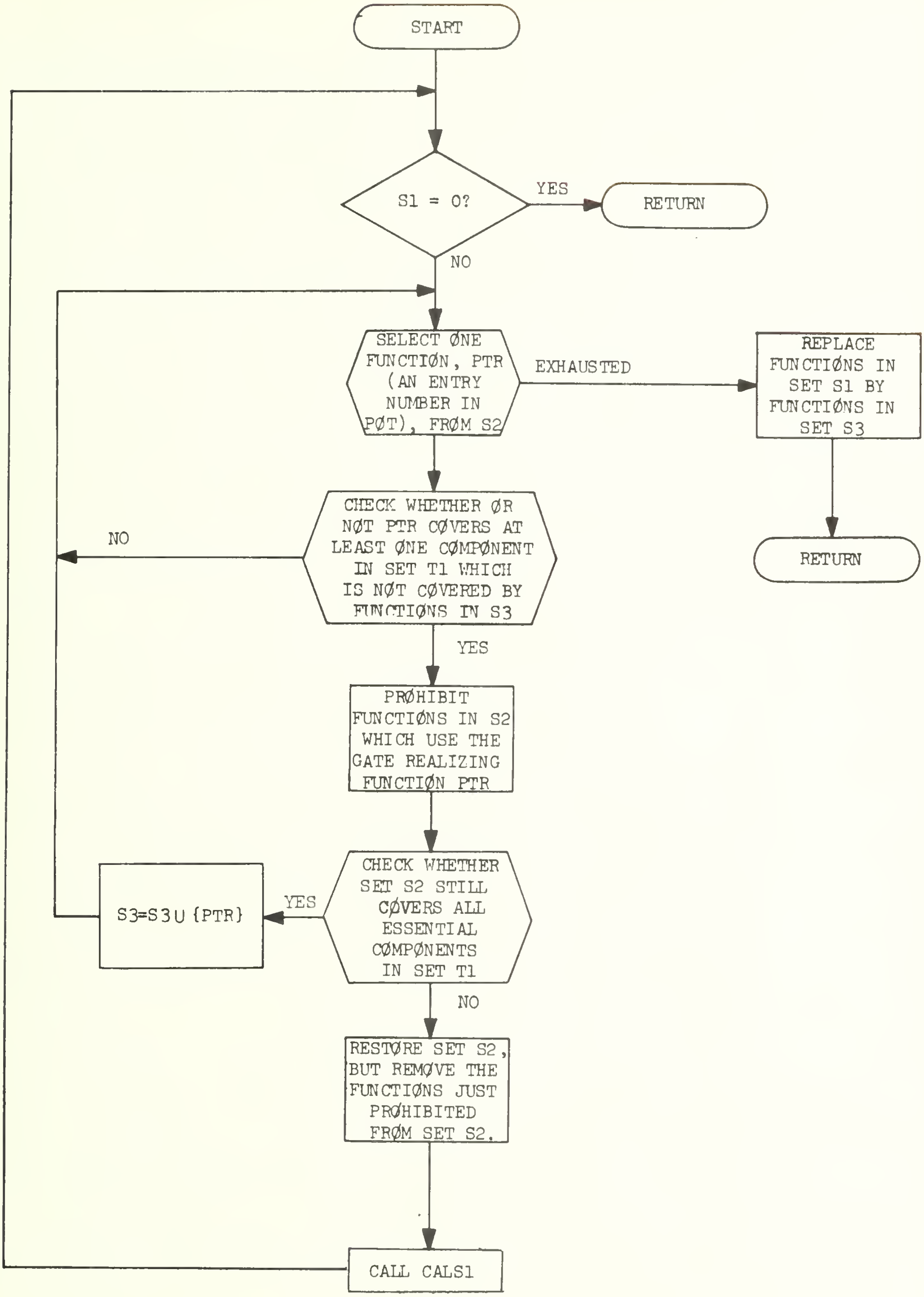

START
S1 = 0?
YES
RETURN
NO
SELECT ØNE FUNCTIØN, PTR (AN ENTRY NUMBER IN PØT), FRØM S2
EXHAUSTED
REPLACE FUNCTIØNS IN SET S1 BY FUNCTIØNS IN SET S3
RETURN
CHECK WHETHER ØR NØT PTR CØVERS AT LEAST ØNE CØMPØNENT IN SET T1 WHICH IS NØT CØVERED BY FUNCTIØNS IN S3
NO
YES
PRØHIBIT FUNCTIØNS IN S2 WHICH USE THE GATE REALIZING FUNCTIØN PTR
CHECK WHETHER SET S2 STILL CØVERS ALL ESSENTIAL CØMPØNENTS IN SET T1
YES
S3=S3∪{PTR}
NO
RESTØRE SET S2, BUT REMØVE THE FUNCTIØNS JUST PRØHIBITED FRØM SET S2.
CALL CALS1

Figure 2.3.2 Flowchart of subroutine RPLCF.

Subroutine ORDRQ2 classifies the input functions except external variables of the selected gate GI into three groups. The first group contains input functions which have no 1-components corresponding to 0-error-components in the CSPFE of GI. The second group contains the functions which have at least one 1-component corresponding to a primary 0-error-component in the CSPFE of GI. The functions which have 1-components corresponding to some 0-error components but none of them are primary error-components are classified as the third group. Since the functions in the first group are not subjects of substitution, they are not listed. The other two groups are put into a 2-dimensional array working as two 1-dimensional arrays, each of which contains the sorted list of the functions in second and third groups, respectively. The sorting is based on the number of 1-components corresponding to the 0-error-components in the CSPFE of GI. These two lists are referenced by subroutine RCEC as the lists of candidates to be replaced.

Subroutine CONECT(PTR) realizes a simple procedure which connects the PTR-th function in the POT to gate GI, the gate under consideration. As the realization of the PTR-th function in the POT may require additional connections as explained in Section 2.2.2, subroutine CONECT also connects these connections.

2.3.3 Propagation of CSPFE's

As discussed in Section 2.3.1, if no actual substitution has taken place and no error-component has been compensated during the error-compensation concerning a selected gate GI, the procedure will propagate the CSPFE of GI to the input connections of GI and will select another gate.

The propagation of CSPFE's is a procedure for calculating CSPFE's. As explained in Section 2.1 of [5], the CSPFE of a particular gate depends on the CSPFE's of its output connections, and the operation for calculating the CSPFE of a gate from the CSPFE's of its output connections is commutative (i.e., the order in which connections are considered is not important). In the procedure realized by subroutine RCEC, the CSPFE for each connection is not stored. Instead, the intermediate CSPFE for each gate is stored, and it will remain an intermediate CSPFE for that gate until all its output connections have been considered. The propagation of the CSPFE of gate GI is essentially the calculation of CSPFE's of its immediate predecessors contributed by gate GI.

Since there are five types of components in the CSPFE, the propagation can be classified into the following five categories.

(1) Don't-care components (*)

The corresponding input components can be either 0 or 1. Since it is a don't-care component, no change is required for the intermediate CSPFE's of the immediate predecessors of gate GI.

(2) 1-components

A 1-component is realized when all corresponding input components are 0. Since the 1-component is required for the CSPFE of gate GI, all the corresponding components in the CSPFE's of GI's immediate predecessors should be assigned to 0 unless the input is from an output gate and the corresponding component has been assigned as a 0-error-component.

(3) 1-error-components

Since a 1-error-component in the CSPFE of GI indicates the actual output is 1 but the preferred value is 0, it should be propagated to its immediate predecessors as 0-error-components if possible. If the corresponding component in the intermediate CSPFE of one of GI's immediate predecessors has already been assigned to 0 before this propagation, it should remain 0 since the assigned 0 means that an earlier propagation from another gate requires that it be 0.

(4) 0-error-components

In this case, the corresponding components of the inputs of GI could be either 0 or 1. If the component of an input corresponding to a 0-error-component in the CSPFE of GI is 0, the corresponding component of the CSPFE of that input is assigned to 0 in order not to increase the degree of the 0-error-component under consideration. On the other hand, if the component corresponding to a 0-error-component is a 1, it should be assigned as a 1-error-component unless it has already been assigned as a 1-component in an earlier step.

(5) 0-components

The propagation of 0-components is the most complicated case among those of five different types of components. For each 0-component in the CSPFE of GI, only one corresponding component of GI's inputs should be assigned to 1, and all others need not change. If a 0-component is covered by an external variable or a gate whose corresponding component in its CSPFE has already been assigned to 1, the above condition has already been satisfied and therefore no other changes are necessary. Since the error-compensation procedure depends very much on how to choose the 1-component to cover a 0-component, the ordering is carefully calculated according to the following.

(a) Inputs which covers fewer primary 0-error-components in the

CSPFE of GI have higher priority in the selection.

(b) If there is a tie based on (a), the input which covers fewer 0-error-components has the higher priority in the selection.

(c) If two inputs have the same priority based on (a) and (b), the one which has more output connections has the higher priority.

Rules (a) and (b) are based on the consideration of easier error-compensation for the immediate predecessors of GI. Since a primary 0-error-component in the CSPFE of GI can be compensated if the 1-error-component covering it is compensated, every effort should be made to make this 1-error-component easier to be compensated. Rules (a) and (b) tend to assign more don't-care-components to the inputs which cover more primary 0-error-components. This explains rules (a) and (b) since the more don't-care-components there are in the CSPFE of a gate, the more connectable functions are likely, and therefore the more chances there are to compensate for these error-components. Rule (c) is based on the number of output connections. If a gate which has more output connections than others is selected and assigned to 1, this 1 is likely to cover 0-components in other immediate successors of this gate.

In the actual program, the priority depends on the weight associated with each input connection which is calculated according to the following formula.

$$\begin{aligned} W = 10^6 - 10^4 * (\text{\# primary error-components}) \\ - 10^2 * (\text{\# error-components}) + (\text{\# immediate successors}) \end{aligned}$$

A special case should be noted. If a 0-component is covered only by an output gate of the network and the corresponding component in the CSPFE of that gate has already been assigned as a 1-error-component, this component should not be changed and the corresponding components of other inputs should be assigned as 0-error-components, if possible, in order to cover the 0-component in the CSPFE of GI when the corresponding 1-error-component in the CSPFE of the output gate is compensated.

2.3.4 Algorithm of error-compensation procedure and flowchart

As a summary of the above discussions, a brief description of the procedure realized by subroutine RCEC is presented in this section. For the detail of the procedure, however, see the flowchart in Figure 2.3.4.1.

The Procedure of Error Compensation Realized by RCEC

Step 1. Selection of gate

Select a gate GI according to the ascending order of the level number assigned to each gate, i.e., a gate in the lowest level is selected first. If all gates have been considered, return (RETURN 1) to the calling subroutine (PROCCE in NETTRA-E1, -E2, and -E3)(this is an unsuccessful compensation).

Step 2. Removal of redundant connections

For each immediate predecessor GJ of GI, call subroutine FORC(GJ) to check whether or not the connection from GJ to GI is redundant. If so, remove this connection.

If no error-component is in the CSPFE of GI, go to Step 9.

Step 3. Selection of connectable functions from POT

Select strongly effectively E-connectable functions for GI from the potential output table. If there is no such connectable function for GI, go to Step 9.

Step 4. Classification of connectable functions

Classify connectable functions for GI into four categories: (1) external variables without error-components (set DIO), (2) gates without anticipated error-components (set BIO), (3) gates with anticipated error-components (set BI), and (4) external variables with error-components (set DI). Sort BI according to the number of 0-error-components the function covers.

Step 5. Substituting for external variables with errors

Let S2 = DIO U BIO U BI, S = external variable inputs of GI, which cover 0-error-components (these inputs are found and sorted by calling subroutine ORDRQ2). Call subroutine CALS1 and RPLCF to substitute a subset S3 of S2 for a subset S1 of S, if possible.

Step 6. Substituting for functions covering primary error-components

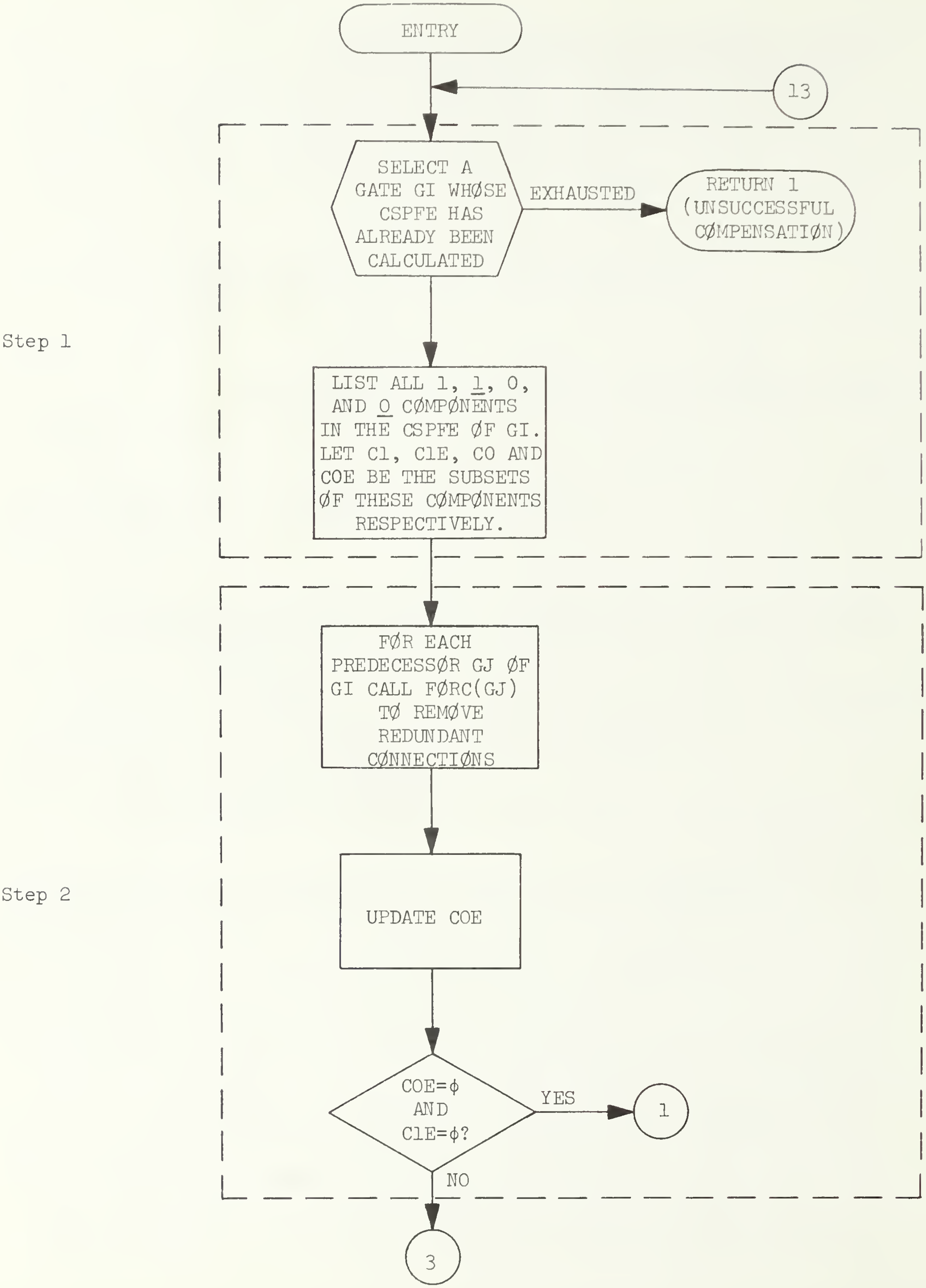
For each input of GI covering primary error-components (these inputs are found and sorted by calling ORDRQ2 in Step 5), check whether or not it can be replaced by functions in DIO, BIO and BI with some primary error-components corrected. Call CALS1 and RPLCF to complete this substitution.

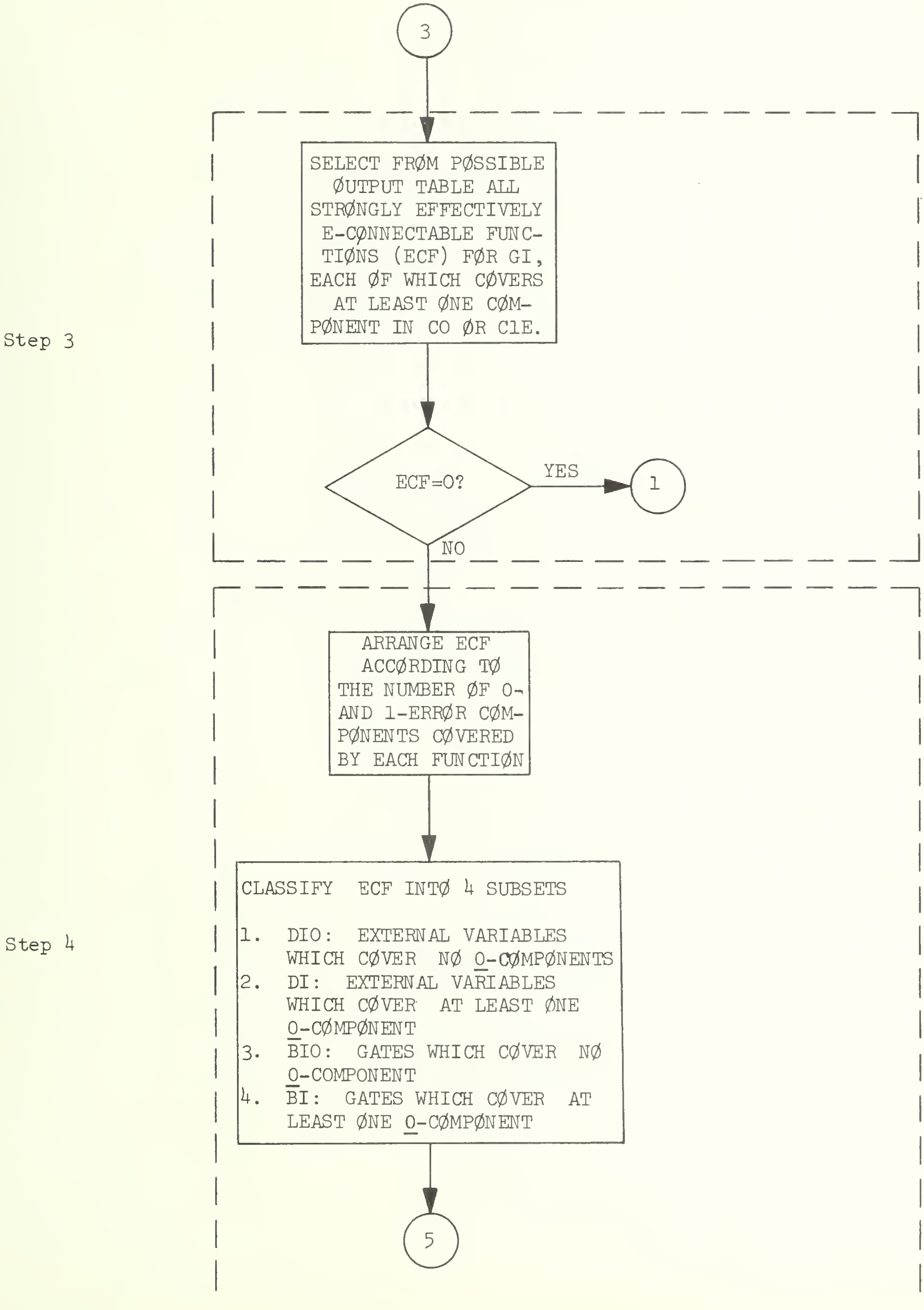
Step 7. Substituting for functions by functions having no error-components

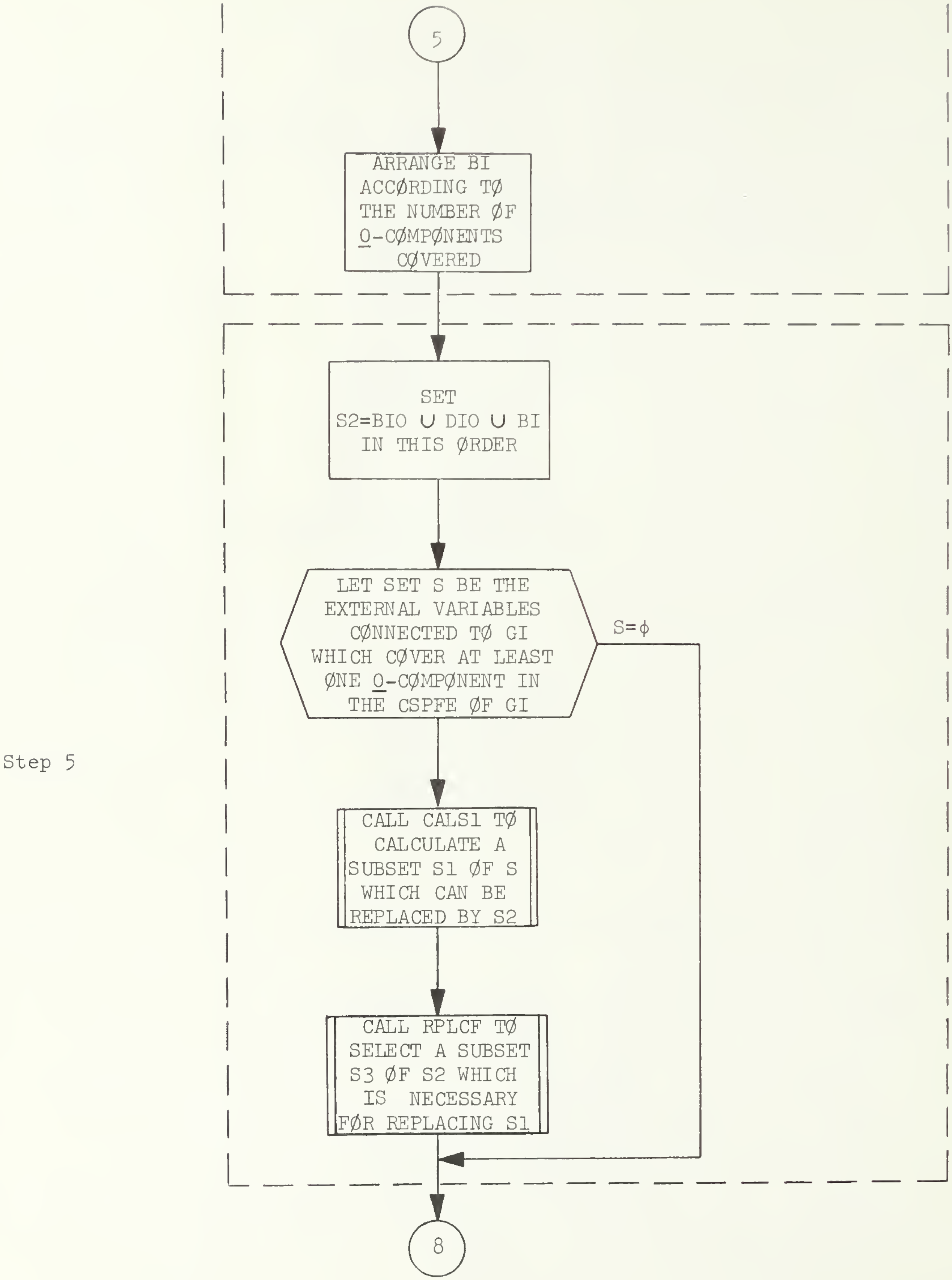
Let S be the set of remaining inputs of GI which have error-components, and let S2 be the set of the remaining functions (exclude already selected and/or prohibited ones) in sets DIO and BIO. Call CALS1 and RPLCF to replace as subset S1 of S by functions in a subset S3 of S2, if possible.

Step 8. Compensating for 1-error-components in the CSPFE of GI

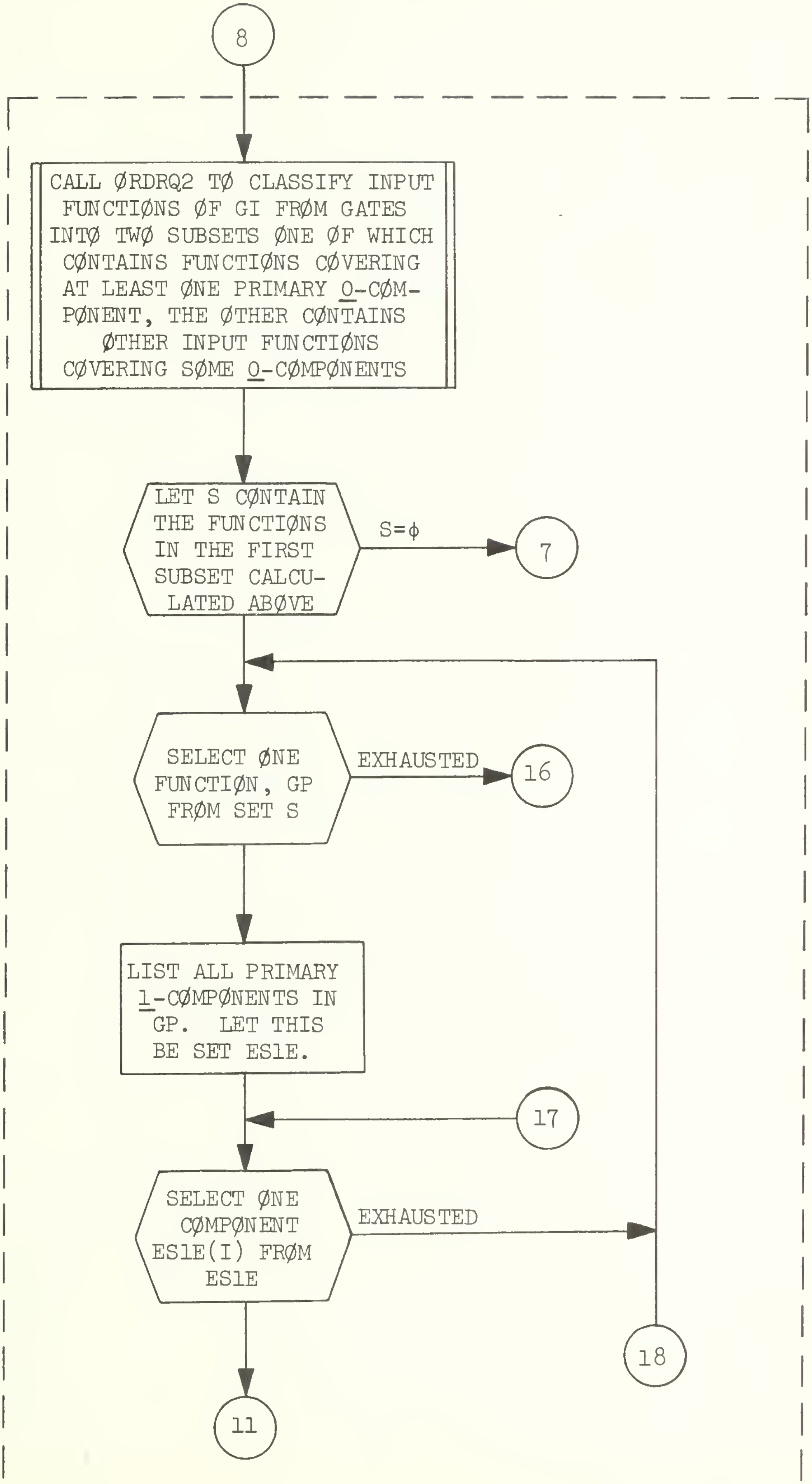
If some of the remaining functions in DIO, BIO and BI have 1-components corresponding to some 1-error-components in the CSPFE of GI,

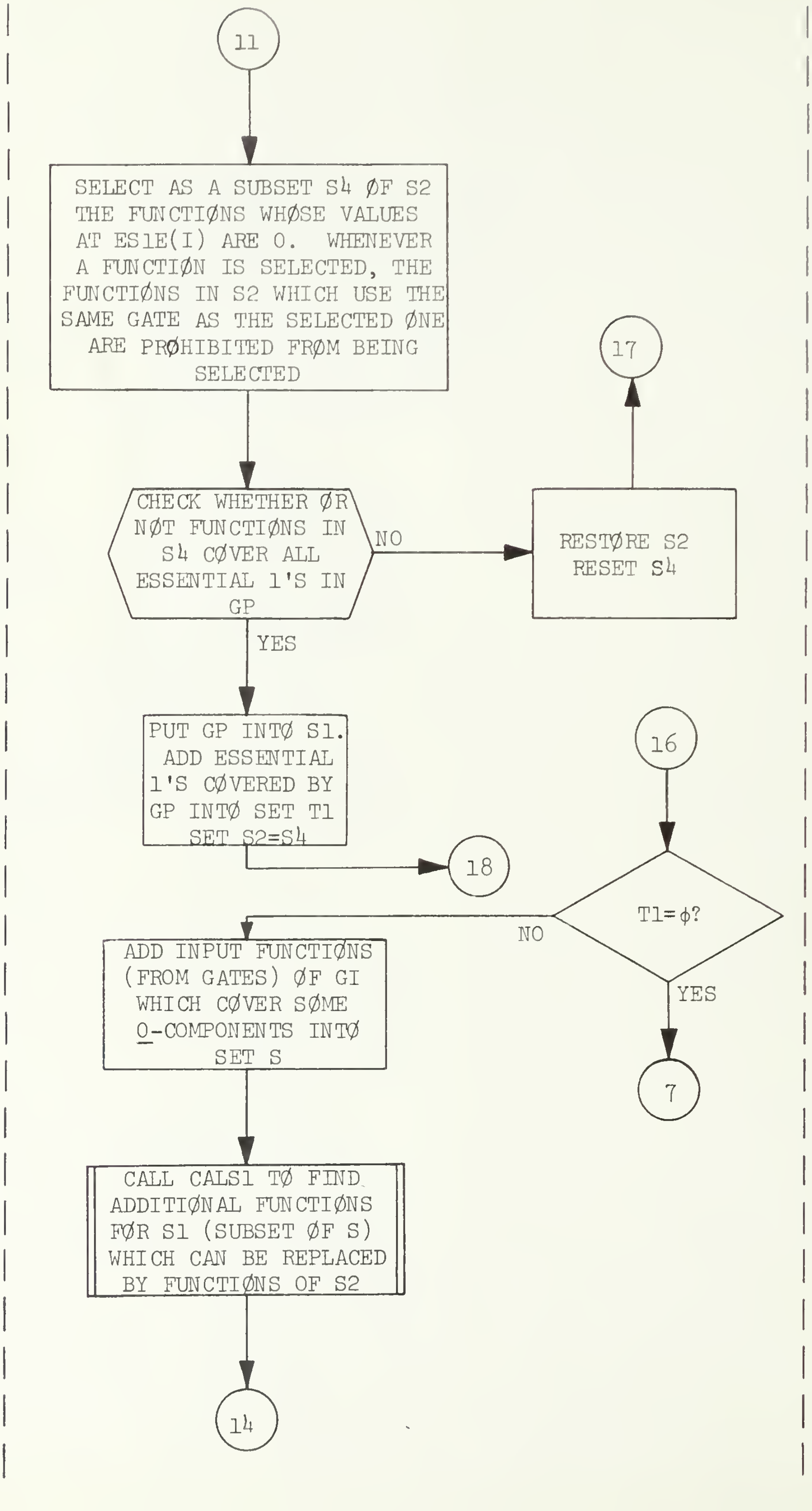

ENTRY
13
SELECT A GATE GI WHØSE CSPFE HAS ALREADY BEEN CALCULATED
EXHAUSTED
RETURN 1 (UNSUCCESSFUL CØMPENSATIØN)
Step 1
LIST ALL 1, 1̲, 0, AND 0̲ CØMPØNENTS IN THE CSPFE ØF GI. LET C1, C1E, CO AND COE BE THE SUBSETS ØF THESE CØMPØNENTS RESPECTIVELY.
FØR EACH PREDECESSØR GJ ØF GI CALL FØRC(GJ) TØ REMØVE REDUNDANT CØNNECTIØNS
Step 2
UPDATE COE
COE=ϕ AND C1E=ϕ?
YES
1
NO
3

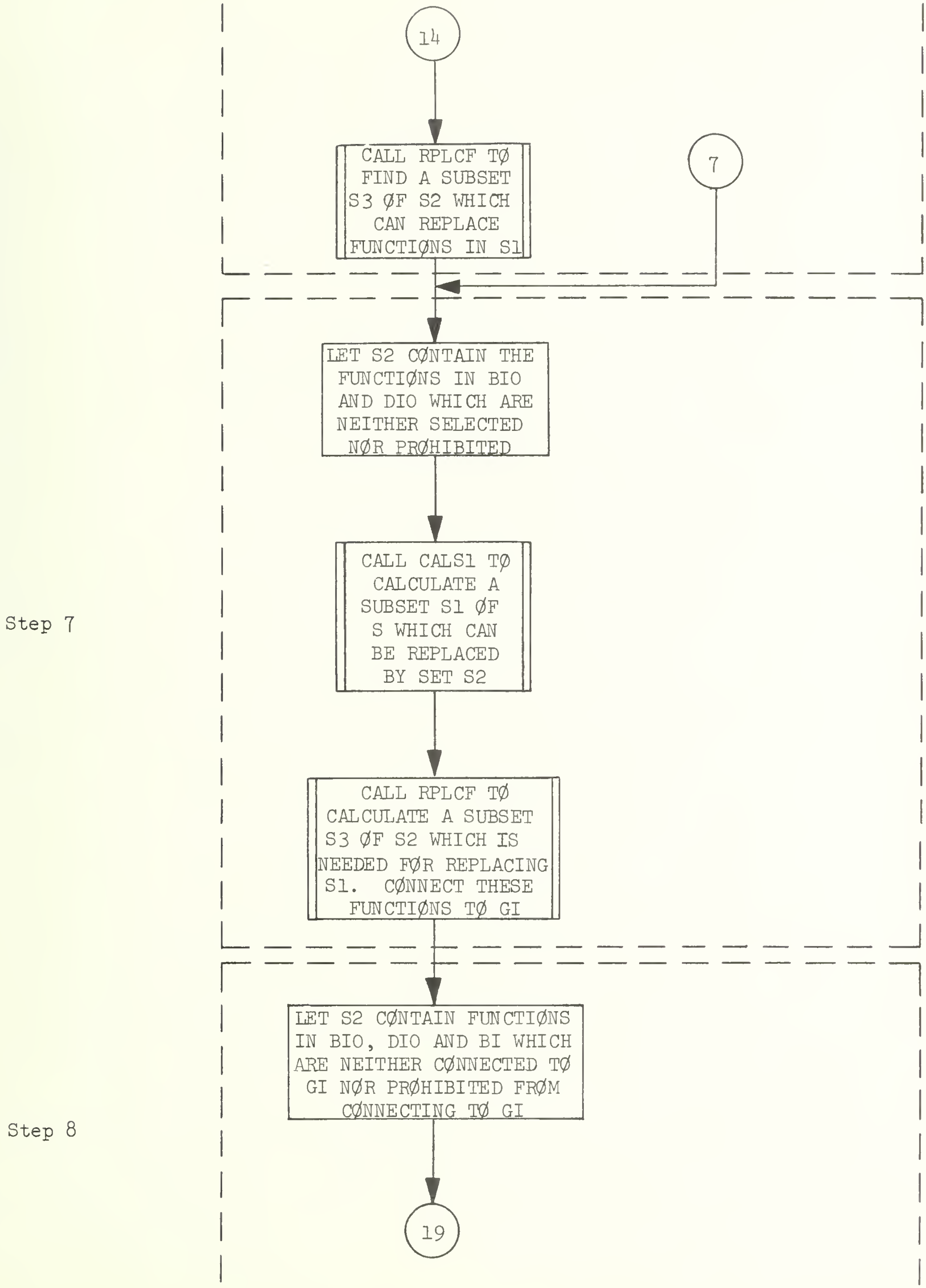

3
SELECT FRØM PØSSIBLE ØUTPUT TABLE ALL STRØNGLY EFFECTIVELY E-CØNNECTABLE FUNCTIØNS (ECF) FØR GI, EACH ØF WHICH CØVERS AT LEAST ØNE CØMPØNENT IN CO ØR C1E.
Step 3
ECF=O?
YES
1
NO
ARRANGE ECF ACCØRDING TØ THE NUMBER ØF 0- AND 1-ERRØR CØMPØNENTS CØVERED BY EACH FUNCTIØN
CLASSIFY ECF INTØ 4 SUBSETS
1. DIO: EXTERNAL VARIABLES WHICH CØVER NØ 0-CØMPØNENTS
2. DI: EXTERNAL VARIABLES WHICH CØVER AT LEAST ØNE 0-CØMPØNENT
3. BIO: GATES WHICH CØVER NØ 0-COMPONENT
4. BI: GATES WHICH CØVER AT LEAST ØNE 0-CØMPØNENT
Step 4
5

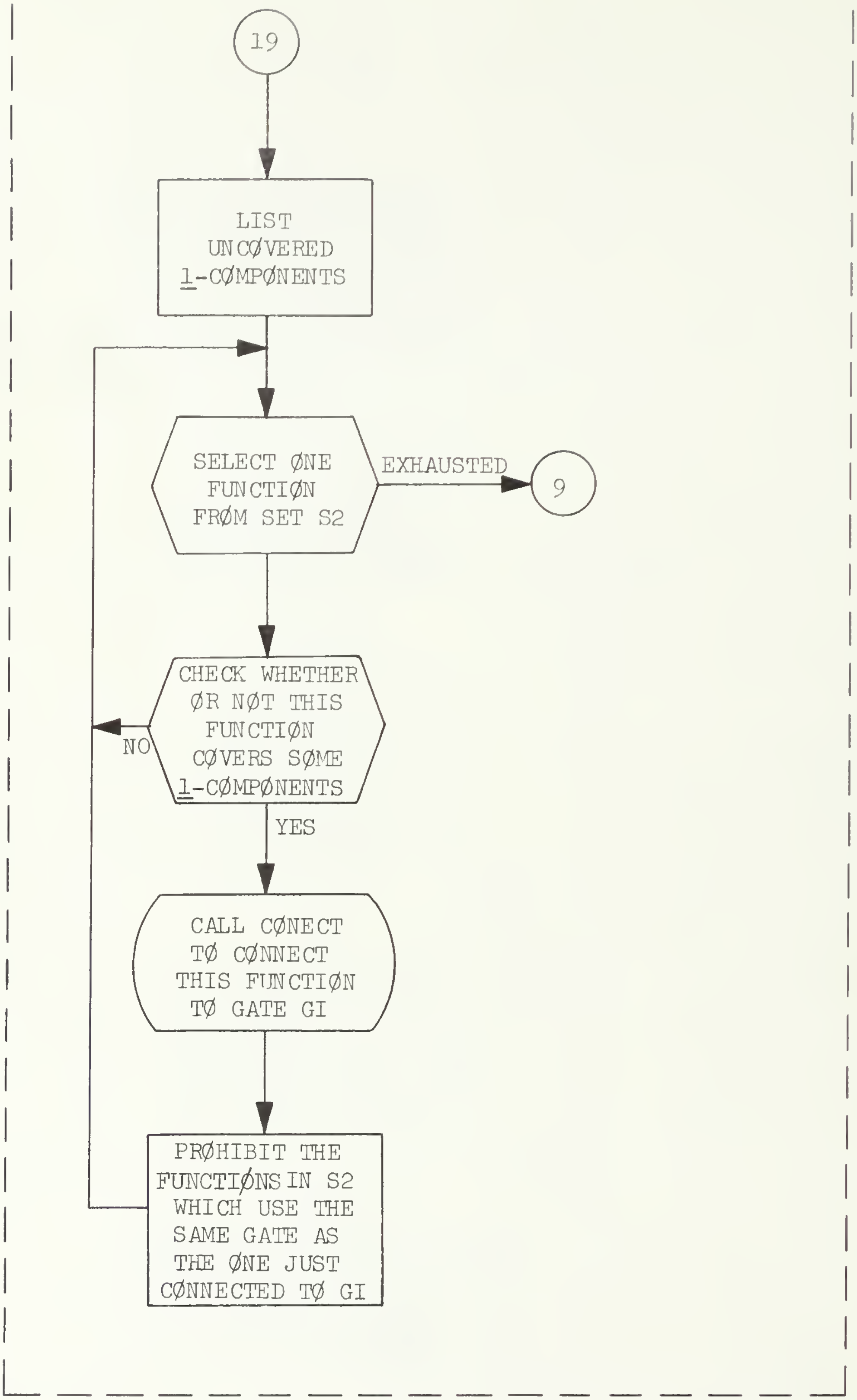

5
ARRANGE BI ACCØRDING TØ THE NUMBER ØF 0-CØMPØNENTS CØVERED
SET S2=BIO ∪ DIO ∪ BI IN THIS ØRDER
LET SET S BE THE EXTERNAL VARIABLES CØNNECTED TØ GI WHICH CØVER AT LEAST ØNE 0-CØMPØNENT IN THE CSPFE ØF GI
S=ϕ
CALL CALS1 TØ CALCULATE A SUBSET S1 ØF S WHICH CAN BE REPLACED BY S2
CALL RPLCF TØ SELECT A SUBSET S3 ØF S2 WHICH IS NECESSARY FØR REPLACING S1
8
Step 5

Step 6

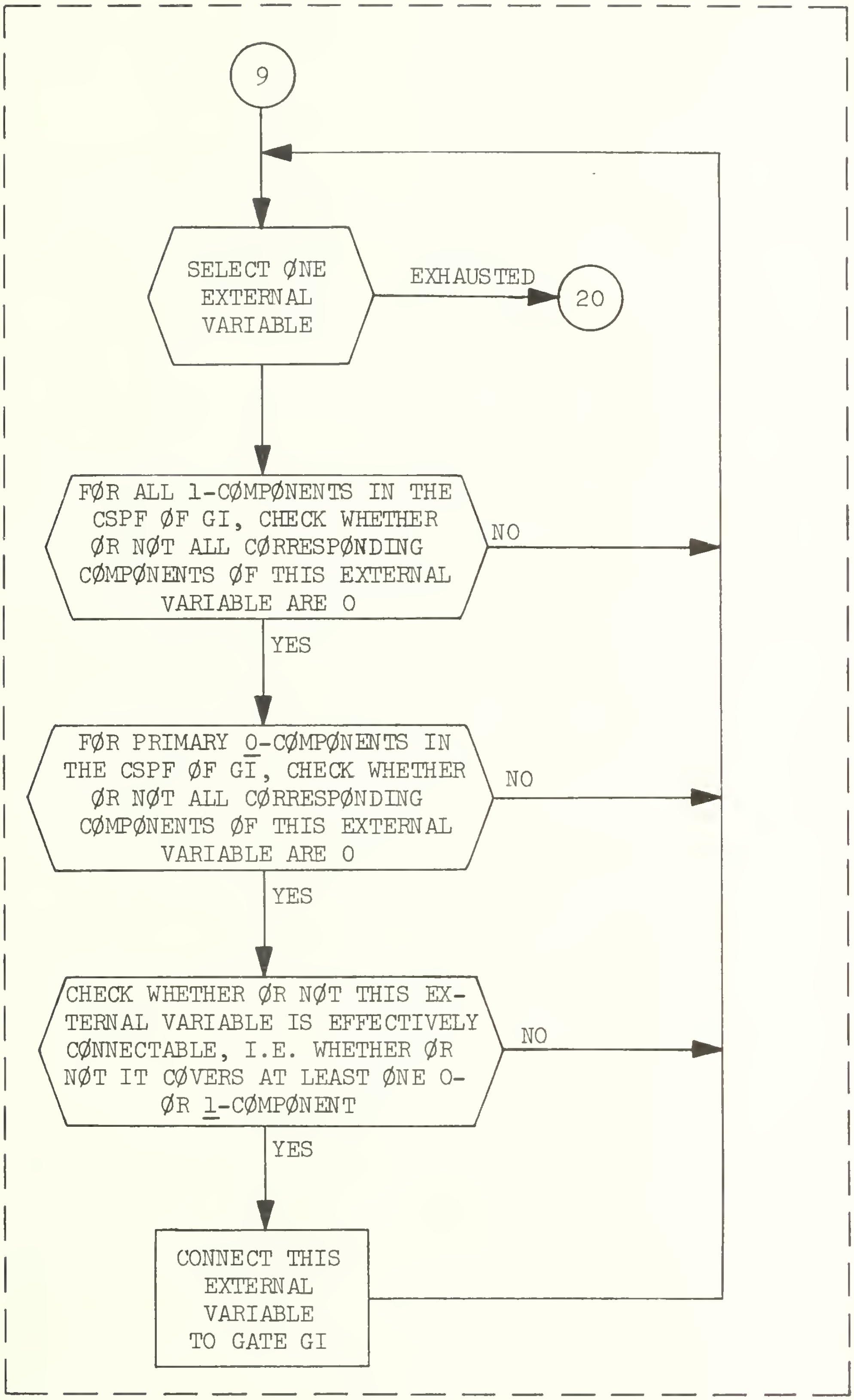

8
CALL ØRDRQ2 TØ CLASSIFY INPUT FUNCTIØNS ØF GI FRØM GATES INTØ TWØ SUBSETS ØNE ØF WHICH CØNTAINS FUNCTIØNS CØVERING AT LEAST ØNE PRIMARY 0-CØMPØNENT, THE ØTHER CØNTAINS ØTHER INPUT FUNCTIØNS CØVERING SØME 0-CØMPØNENTS
LET S CØNTAIN THE FUNCTIØNS IN THE FIRST SUBSET CALCULATED ABØVE
S=ϕ
7
SELECT ØNE FUNCTIØN, GP FRØM SET S
EXHAUSTED
16
LIST ALL PRIMARY 1-CØMPØNENTS IN GP. LET THIS BE SET ES1E.
17
SELECT ØNE CØMPØNENT ES1E(I) FRØM ES1E
EXHAUSTED
18
11
]

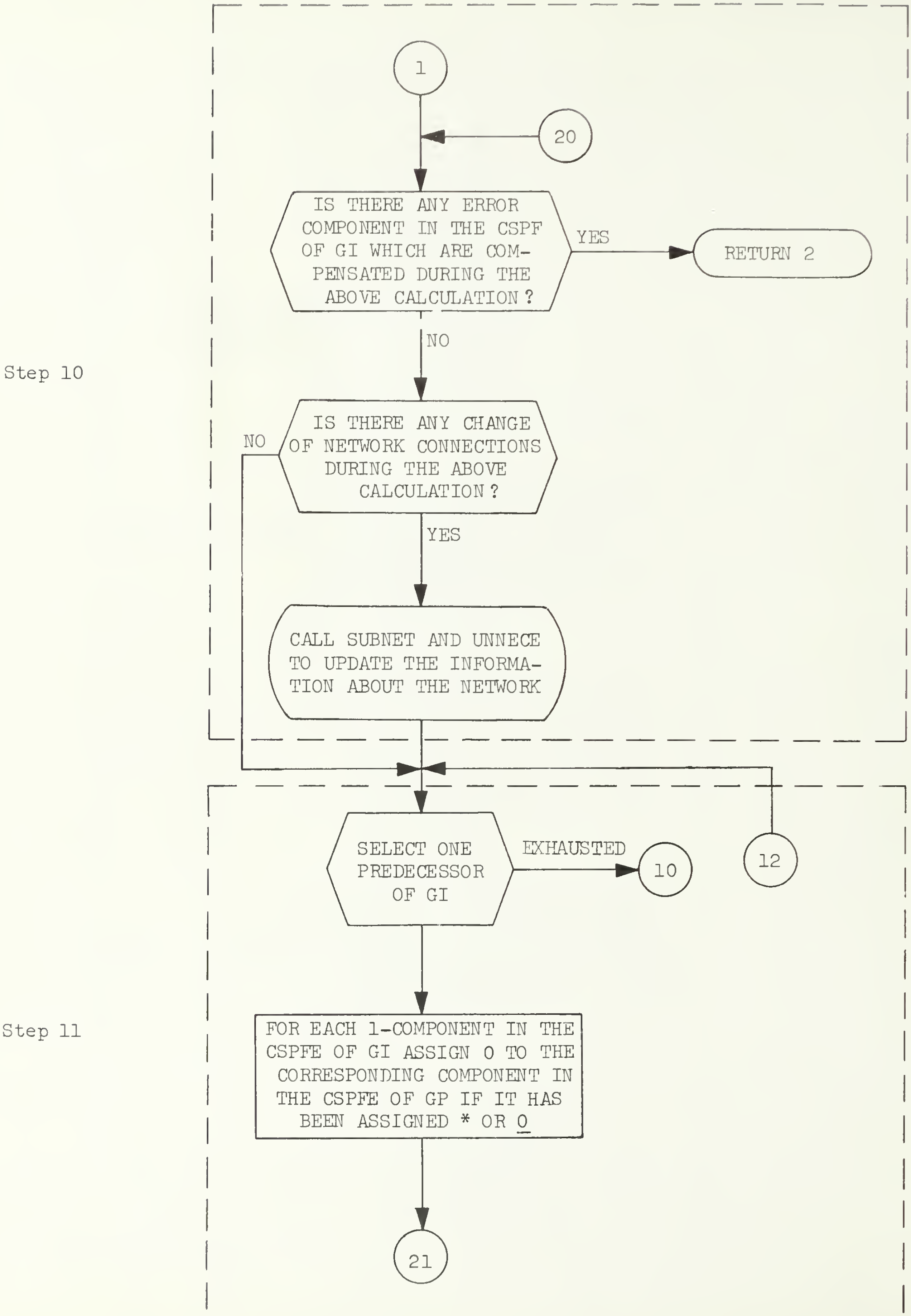

11
SELECT AS A SUBSET S4 ØF S2 THE FUNCTIØNS WHØSE VALUES AT ES1E(I) ARE 0. WHENEVER A FUNCTIØN IS SELECTED, THE FUNCTIØNS IN S2 WHICH USE THE SAME GATE AS THE SELECTED ØNE ARE PRØHIBITED FRØM BEING SELECTED
17
CHECK WHETHER ØR NØT FUNCTIØNS IN S4 CØVER ALL ESSENTIAL 1'S IN GP
NO
RESTØRE S2
RESET S4
YES
PUT GP INTØ S1. ADD ESSENTIAL 1'S CØVERED BY GP INTØ SET T1 SET S2=S4
16
18
T1=ϕ?
NO
ADD INPUT FUNCTIØNS (FROM GATES) ØF GI WHICH CØVER SØME 0-COMPONENTS INTØ SET S
YES
7
CALL CALS1 TØ FIND ADDITIØNAL FUNCTIØNS FØR S1 (SUBSET ØF S) WHICH CAN BE REPLACED BY FUNCTIØNS OF S2
14

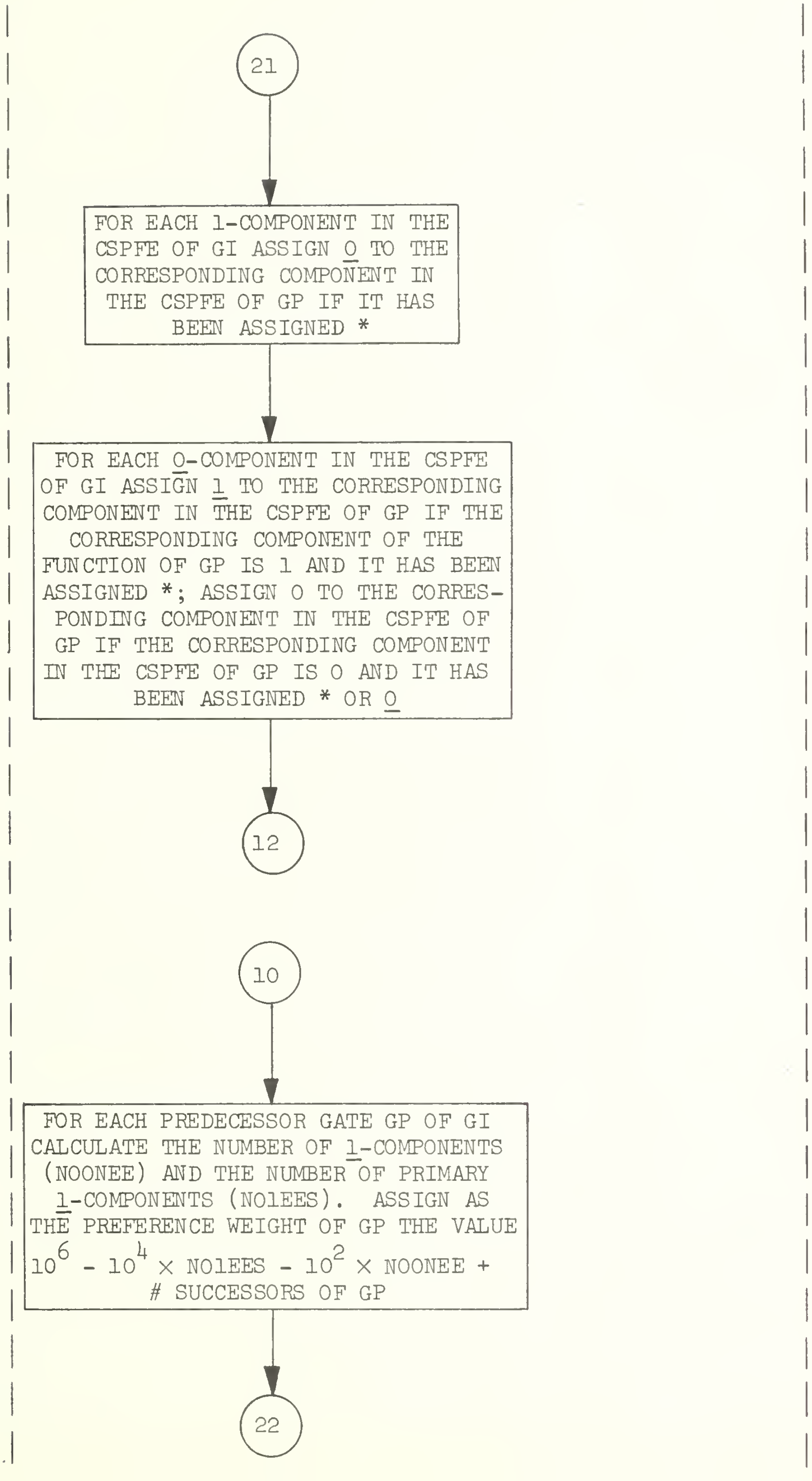

14
CALL RPLCF TØ FIND A SUBSET S3 ØF S2 WHICH CAN REPLACE FUNCTIØNS IN S1
7
LET S2 CØNTAIN THE FUNCTIØNS IN BIO AND DIO WHICH ARE NEITHER SELECTED NØR PRØHIBITED
Step 7
CALL CALS1 TØ CALCULATE A SUBSET S1 ØF S WHICH CAN BE REPLACED BY SET S2
CALL RPLCF TØ CALCULATE A SUBSET S3 ØF S2 WHICH IS NEEDED FØR REPLACING S1. CØNNECT THESE FUNCTIØNS TØ GI
LET S2 CØNTAIN FUNCTIØNS IN BIO, DIO AND BI WHICH ARE NEITHER CØNNECTED TØ GI NØR PRØHIBITED FRØM CØNNECTING TØ GI
Step 8
19

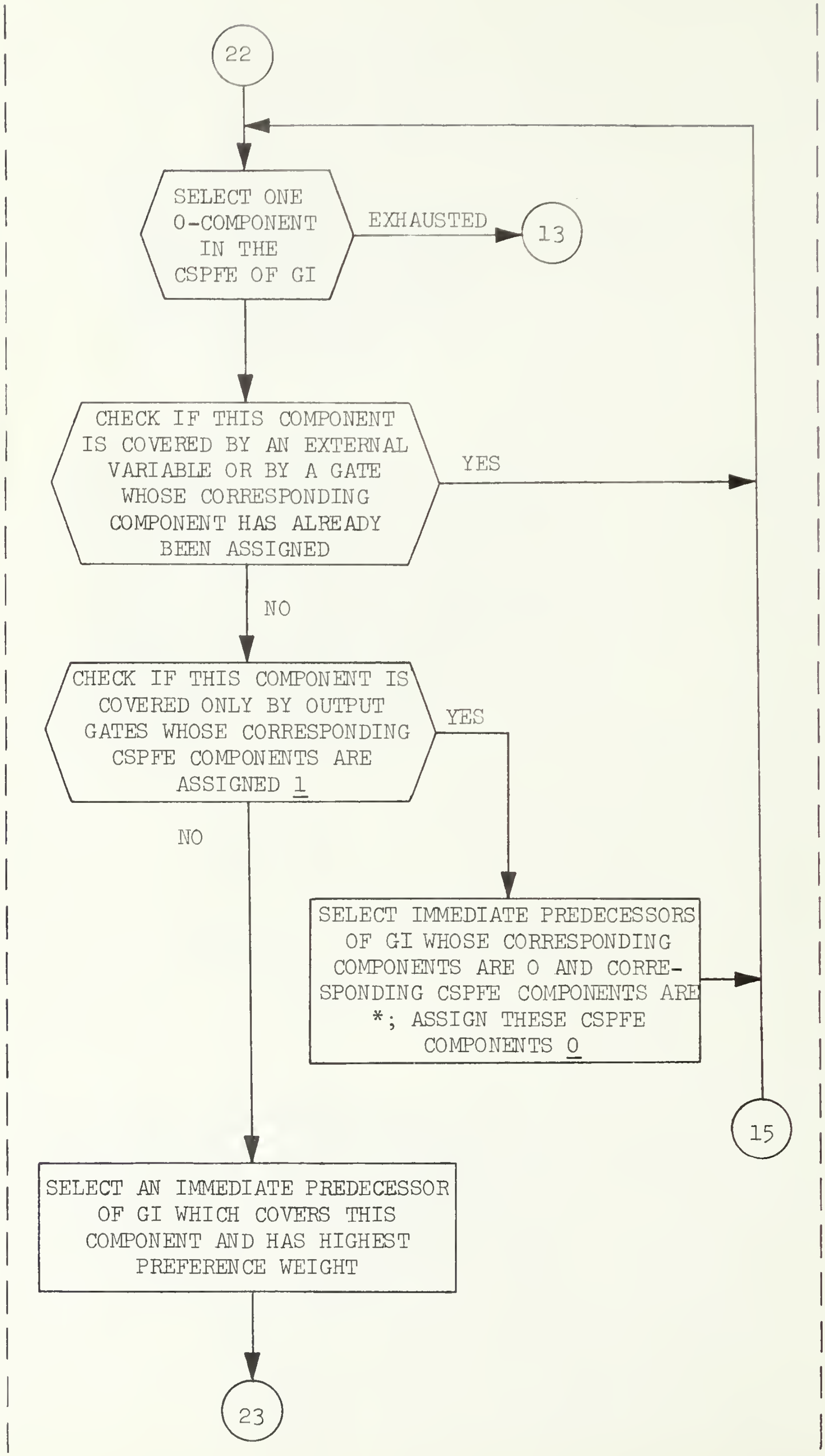

19
LIST UNCØVERED 1-CØMPØNENTS
SELECT ØNE FUNCTIØN FRØM SET S2
EXHAUSTED
9
CHECK WHETHER ØR NØT THIS FUNCTIØN CØVERS SØME 1-CØMPØNENTS
NO
YES
CALL CØNECT TØ CØNNECT THIS FUNCTIØN TØ GATE GI
PRØHIBIT THE FUNCTIØNS IN S2 WHICH USE THE SAME GATE AS THE ØNE JUST CØNNECTED TØ GI

Step 9


9
SELECT ØNE EXTERNAL VARIABLE
EXHAUSTED
20
FØR ALL 1-CØMPØNENTS IN THE CSPF ØF GI, CHECK WHETHER ØR NØT ALL CØRRESPØNDING CØMPØNENTS ØF THIS EXTERNAL VARIABLE ARE 0
NO
YES
FØR PRIMARY 0-CØMPØNENTS IN THE CSPF ØF GI, CHECK WHETHER ØR NØT ALL CØRRESPØNDING CØMPØNENTS ØF THIS EXTERNAL VARIABLE ARE 0
NO
YES
CHECK WHETHER ØR NØT THIS EXTERNAL VARIABLE IS EFFECTIVELY CØNNECTABLE, I.E. WHETHER ØR NØT IT CØVERS AT LEAST ØNE 0- ØR 1-CØMPØNENT
NO
YES
CONNECT THIS EXTERNAL VARIABLE TO GATE GI


1
20
IS THERE ANY ERROR COMPONENT IN THE CSPF OF GI WHICH ARE COMPENSATED DURING THE ABOVE CALCULATION?
YES
RETURN 2
NO
Step 10
IS THERE ANY CHANGE OF NETWORK CONNECTIONS DURING THE ABOVE CALCULATION?
NO
YES
CALL SUBNET AND UNNECE TO UPDATE THE INFORMATION ABOUT THE NETWORK
SELECT ONE PREDECESSOR OF GI
EXHAUSTED
10
12
Step 11
FOR EACH 1-COMPONENT IN THE CSPFE OF GI ASSIGN 0 TO THE CORRESPONDING COMPONENT IN THE CSPFE OF GP IF IT HAS BEEN ASSIGNED * OR 0
21


21
FOR EACH 1-COMPONENT IN THE CSPFE OF GI ASSIGN 0 TO THE CORRESPONDING COMPONENT IN THE CSPFE OF GP IF IT HAS BEEN ASSIGNED *
FOR EACH 0-COMPONENT IN THE CSPFE OF GI ASSIGN 1 TO THE CORRESPONDING COMPONENT IN THE CSPFE OF GP IF THE CORRESPONDING COMPONENT OF THE FUNCTION OF GP IS 1 AND IT HAS BEEN ASSIGNED *; ASSIGN 0 TO THE CORRESPONDING COMPONENT IN THE CSPFE OF GP IF THE CORRESPONDING COMPONENT IN THE CSPFE OF GP IS 0 AND IT HAS BEEN ASSIGNED * OR 0
12
10
FOR EACH PREDECESSOR GATE GP OF GI CALCULATE THE NUMBER OF 1-COMPONENTS (NOONEE) AND THE NUMBER OF PRIMARY 1-COMPONENTS (NO1EES). ASSIGN AS THE PREFERENCE WEIGHT OF GP THE VALUE $10^6 - 10^4 \times$ NO1EES $- 10^2 \times$ NOONEE + # SUCCESSORS OF GP
22


22
SELECT ONE 0-COMPONENT IN THE CSPFE OF GI
EXHAUSTED
13
CHECK IF THIS COMPONENT IS COVERED BY AN EXTERNAL VARIABLE OR BY A GATE WHOSE CORRESPONDING COMPONENT HAS ALREADY BEEN ASSIGNED
YES
NO
CHECK IF THIS COMPONENT IS COVERED ONLY BY OUTPUT GATES WHOSE CORRESPONDING CSPFE COMPONENTS ARE ASSIGNED 1
YES
NO
SELECT IMMEDIATE PREDECESSORS OF GI WHOSE CORRESPONDING COMPONENTS ARE 0 AND CORRESPONDING CSPFE COMPONENTS ARE *; ASSIGN THESE CSPFE COMPONENTS 0
15
SELECT AN IMMEDIATE PREDECESSOR OF GI WHICH COVERS THIS COMPONENT AND HAS HIGHEST PREFERENCE WEIGHT
23

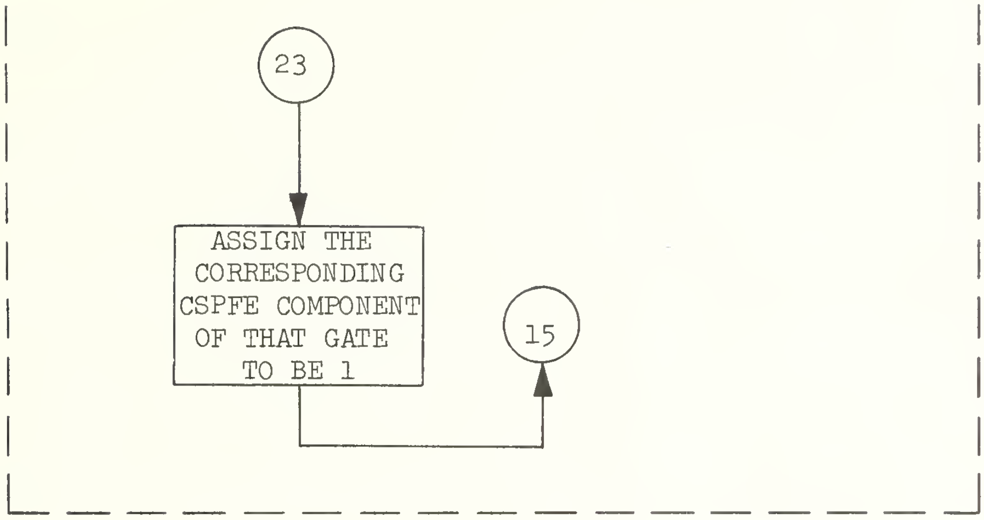

23
ASSIGN THE CORRESPONDING CSPFE COMPONENT OF THAT GATE TO BE 1
15

Figure 2.3.4.1 Flowchart of Subroutine RCEC (Dashed blocks are the steps 1, 2, ..., 11 corresponding to those of the error compensation procedure).

connect these functions to compensate for 1-error-components.

Step 9. Adding redundant external variables

Connect connectable redundant external variables to GI without reducing the number of primary error-components in the CSPFE of GI.

Step 10. Return to calling subroutine

If an error-component in the CSPFE of GI has been corrected or a substitution is performed during steps 2, 5, 6, 7 or 8, return (RETURN 2) to the calling subroutine (PROCCE in NETTRA-E1, -E2, and -E3) to check the outputs of the network.

Step 11. Propagation of CSPFE

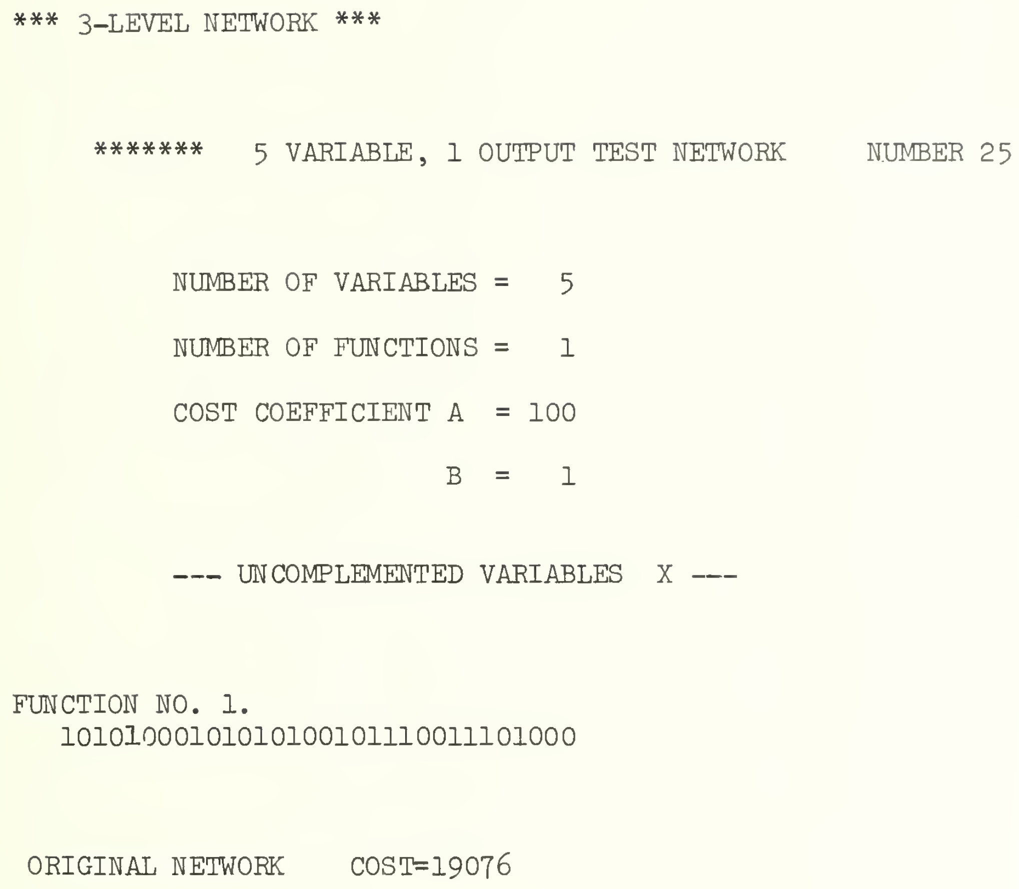
Update the intermediate CSPFE for each immediate predecessor of gate GI. Go to Step 1.

The relation between subroutine RCEC and its supporting subroutines is shown in Figure 2.1 (in Section 2).

2.4 Example for NETTRA-E1

The printout obtained during the solution of a typical problem by NETTRA-E1 is shown in Figure 2.4.1. The original network, as specified in the beginning of the printout (Figure 2.4.1(a)), consists of 19 gates and 76 connections and realizes a single 5-variable output function. Only uncomplemented variables are available as inputs to the network.

Following this information is printed a complete truth table (b) showing the output of every gate in the original network for every possible input combination. Note that it is gate 1 which realizes the output function of the network.



```
*** 3-LEVEL NETWORK ***


     *******   5 VARIABLE, 1 OUTPUT TEST NETWORK     NUMBER 25


          NUMBER OF VARIABLES =   5

          NUMBER OF FUNCTIONS =   1

          COST COEFFICIENT A  = 100

                           B  =   1

          --- UNCOMPLEMENTED VARIABLES  X ---


FUNCTION NO. 1.
   10101000101010100101110011101000


 ORIGINAL NETWORK    COST=19076
```

(a) Heading information and network parameters.

Figure 2.4.1 Printout obtained from NETTRA-E1 for a sample problem.

TRUTH TABLE

```
X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
X2 = 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
X3 = 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 1 = 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 1 0 1 1 1 0 0 1 1 1 0 1 0 0 0
 2 = 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 3 = 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 4 = 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 5 = 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 6 = 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 7 = 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 8 = 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 9 = 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 = 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
12 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
13 = 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14 = 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
15 = 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0
16 = 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0
17 = 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0
18 = 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
19 = 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1
```

(b) Truth table for original network.

```
GATE .. LEVEL      FED BY


  1     / 1/        3   5   7   8  10  11  12  19

  2     / 3/       X1  X2  X3  X4  X5

  3     / 2/       X1  X2  X3  X4   2

  4     / 3/       X1  X2  X3  X5

  5     / 2/       X1  X2  X3   4

  6     / 3/       X1  X2  X4  X5

  7     / 2/       X1  X2  X4   6

  8     / 2/       X1  X2  X5   2   4   6

  9     / 3/       X1  X3  X4  X5

 10     / 2/       X1  X3  X4   9

 11     / 2/       X2  X3  X4  X5   2

 12     / 2/       X2  X3  X5   2   4

 13     / 3/       X1  X5

 14     / 3/       X2  X3

 15     / 3/       X2  X4

 16     / 3/       X3  X4

 17     / 3/       X3  X5

 18     / 3/       X4  X5

 19     / 2/       13  14  15  16  17  18
```

(c) Configuration of original network.

```
**** BEGIN   1-TH APPLICATION OF PROCCE :          *************

         NETWORK DERIVED BY PROCCE
         TIME ELAPSED =      14  CENTISECONDS

   TRUTH TABLE

   X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

   X2 = 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

   X3 = 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1

   X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1

   X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

    1 = 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 1 0 1 1 1 0 0 1 1 1 0 1 0 0 0

    2 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

    3 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

    4 = 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

    5 = 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

    6 = 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0

    7 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

    8 = 0 1 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

    9 = 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0

   10 = 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

   11 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

   12 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

   13 = 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

   14 = 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0

   15 = 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0

   16 = 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0

   17 = 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0

   18 = 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0

   19 = 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1
```

(d) Truth table for transformed, reduced network.

```
GATE .. LEVEL      FED BY

   1     / 1/        5   8  10  12  19

   2     / 1/

   3     / 1/

   4     / 3/       X1  X3  X5

   5     / 2/       X1  X2  X3   4

   6     / 3/       X4  X5

   7     / 1/

   8     / 2/       X1  X2   4   6

   9     / 3/       X5

  10     / 2/       X1  X3  X4   9

  11     / 1/

  12     / 2/       X2  X3  X5   4

  13     / 3/       X1  X5

  14     / 3/       X2  X3

  15     / 3/       X2  X4

  16     / 3/       X3  X4

  17     / 3/       X3  X5

  18     / 3/       X4  X5

  19     / 2/       13  14  15  16  17  18

* A NETWORK DERIVED BY PROCCE
  COST=15045.
```

(e) Configuration of transformed, reduced network.

Next appears a description of the configuration of the network (c). Each gate is listed along with the gates and/or external variables which are its inputs. The level numbers, which can also be seen in (c), will be discussed later in Section 5.3.

The truth table (note that the outputs for disconnected gates are shown as all 1's) and network configuration for the transformed, reduced network resulting from the action of NETTRA-E1 are shown in (d) and (e), respectively. The derived network, found in .14 seconds, consists of 15 gates and 45 connections. If NETTRA-E1 were applied to this new network, a network with even fewer gates and connections would be obtained.

3. REPETITIVE APPLICATION OF ERROR-COMPENSATION PROCEDURE

Now that it has been explained how the error-compensation procedure (as used in NETTRA-E1) is often able to remove a gate from a network (containing an excessive number of gates) and alter the remaining network configuration so as to again produce the desired functions, the most obvious extension can be made to produce an even better program to reduce the number of gates in a network. If the procedure can remove one gate, it may be able to remove a second, or a third. Thus the programs NETTRA-E2 and -E3 are introduced in the next sections: 3.1 and 3.2.

3.1 Single-Path Application

The storage requirements of NETTRA-E2 are almost identical to those given for NETTRA-E1 since the programs only differ by a few FORTRAN statements in the subroutine MAIN. NETTRA-E2 requires 163K bytes of core storage, about 78K for the actual program instructions and about 85K for the stored data.

The following subroutines, written in FORTRAN IV for the IBM 360/75, constitute the program NETTRA-E2: CALS1, CONECT, FORC, MAIN (this differs from the MAIN in NETTRA-E1), MINI2, ORDRQ2, OUTPUT, POT, PROCCE, RCEC, RPLCF, and SUBNET. Two system-supplied timing subroutines, STIMEZ and KTIMEZ are also assumed to be available, but if they are not, their use can be omitted from the program, or another suitable timing routine substituted, without harming the procedure itself.

The general organization of the program is identical to that of NETTRA-E1. It is shown in Figure 2.1. In this figure, an arrow from block i to block j represents the fact that the subroutine in block i calls the subroutine in block j.

It is subroutine MAIN which actually performs the repeated calls to subroutine PROCCE, removing one gate after another until PROCCE can no longer produce a reduced network.(i.e., a network with fewer gates).

The setup of the input data for NETTRA-E2 is the same as the setup for NETTRA-E1 and -E3. The details can be found in section 5. Listings of all of the subroutines used in NETTRA-E2 can be located in the appendix.

3.1.2 Example of NETTRA-E2

Beginning with the same initial network described by Figure 2.4.1 (a), (b), and (c), NETTRA-E2 is applied. Since NETTRA-E2 is essentially just repetitive applications of NETTRA-E1, the first portion of the printout obtained by NETTRA-E2 is identical to what appears in Figure 2.4.1 (a) - (e).

Following that, PROCCE is called a second, third, fourth and fifth time, each time finding a transformed, reduced network of fewer gates than before (except that the fifth result is not improved over the fourth) and printing out the new truth table and network configuration (as in Figure 2.4.1 (d) and (e)).

For this example, NETTRA-E2 could not find a better network than that obtained in the fourth application of PROCCE. The truth table for this new network is given in Figure 3.1.2.1 (a) and the transformed, reduced network configuration description is shown in Figure 3.1.2.1 (b) as they appear in the

```
****  BEGIN   4-TH APPLICATION OF PROCCE :          ************

          NETWORK DERIVED BY PROCCE
          TIME ELAPSED =       94  CENTISECONDS

 TRUTH TABLE

 X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 X2 = 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

 X3 = 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1

 X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1

 X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

  1 = 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 1 0 1 1 1 0 0 1 1 1 0 1 0 0 0

  2 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

  3 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

  4 = 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

  5 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

  6 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

  7 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

  8 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

  9 = 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

 10 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

 11 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 12 = 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0

 13 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 14 = 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0

 15 = 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0

 16 = 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0

 17 = 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0

 18 = 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0

 19 = 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1
```

(a) Truth table for final network.

Figure 3.1.2.1 Printout of final transformed, reduced network obtained by NETTRA-E2.

```
GATE .. LEVEL         FED BY

   1     / 1/          10  12  19

   2     / 1/

   3     / 1/

   4     / 3/          X1  X3

   5     / 1/

   6     / 1/

   7     / 1/

   8     / 1/

   9     / 3/          X1  X5

  10     / 2/          X1   9

  11     / 1/

  12     / 2/          X2  X5   4  18

  13     / 1/

  14     / 4/          X2  X3

  15     / 3/          X2  X4

  16     / 3/          X3  X4

  17     / 3/          X3  X5

  18     / 3/          X4  X5  14

  19     / 2/           9  14  15  16  17  18

* A NETWORK DERIVED BY PROCCE
  COST=11030.
```

(b) Configuration of final network.

printout from NETTRA-E2. This final transformed, reduced network consists of only 11 gates and 30 connections.

3.2 Multi-Path Application

The program NETTRA-E3 represents what is called a multi-path application of the error-compensation procedure. It is only slightly more complicated than NETTRA-E2 (due to the necessity of storing intermediately produced networks in a stack) and employs only one additional subroutine.

NETTRA-E2 only produces a single sequence of networks beginning with the original network specified by the input data and ending with a network from which the error-compensation routine can successfully remove no more gates. Let this sequence of networks be labeled $W_1 \to W_2 \to W_3 \to \cdots \to W_E$, where W_1 is the original network and W_E is the final network. Each W_{i+1} is derived from W_i by the removal of a single gate (although other gates may consequently be removed) and the successful compensation of the resulting errors by the error-compensation routine. Actually, (in general) there are several different networks that can be obtained from each W_i by the removal of and successful compensation for several different gates. NETTRA-E2 settles for the first network, W_{i+1}, that can be obtained from W_i by the removal of a certain gate and the successful compensation of errors, but NETTRA-E3 stores that improved network (W_{i+1}) and continues to search for additional improved networks that can be derived from W_i by removing different gates. These are also stored, in a stack, and NETTRA-E3 will then search for all of the possible improved networks obtainable from those in the stack, storing the new networks in the stack as they are obtained, and so on. Thus NETTRA-E3 will produce a "tree" of solutions (intermediate and final) while NETTRA-E2 only produces a single "string" of solutions (which, incidentally, will be identical to a certain path through the tree produced by NETTRA-E3).

For example, Figure 3.2.1 shows the difference between the (intermediate and final) solutions obtained by NETTRA-E2 and the solutions obtained by NETTRA-E3.

Beginning with W_1 (assumed to be a network of 16 gates) NETTRA-E2 discovers that gate 6 (for example) can successfully be removed from W_1 to produce the 15-gate network W_2. It immediately prints out this result (i.e., the configuration of W_2) and proceeds to "operate" on W_2, trying to remove some gate in order to obtain a network W_3 of 14 gates.

However, beginning with the same original network, W_1, NETTRA-E3 also finds that gate 6 can be successfully removed to produce a 15-gate network (let it be called $W_{2,1}$ this time), but it does not "forget about" W_1 and immediately attempt to transform $W_{2,1}$. Instead, it prints the result, $W_{2,1}$ (along with a message identifying the "parent" [i.e., W_1] of the network), stores it in a stack memory, and continues to search for other 15-gate networks obtainable from W_1 by the removal of different gates (these are represented by $W_{2,2}$ [obtained by removing gate 8 from W_1], $W_{2,3}$ [removing gate 11], and $W_{2,4}$ [gate 15]). These networks, $W_{2,2}$, $W_{2,3}$, and $W_{2,4}$ are also put into the stack. NETTRA-E3 then selects the top network of the stack ($W_{2,4}$ in this example) and attempts to remove individual gates from it in the same manner as it treated W_1. This process continues until the stack is empty. The terminal nodes of the "solution tree" must then be searched to identify the best solution (i.e., the network with the fewest number of gates).

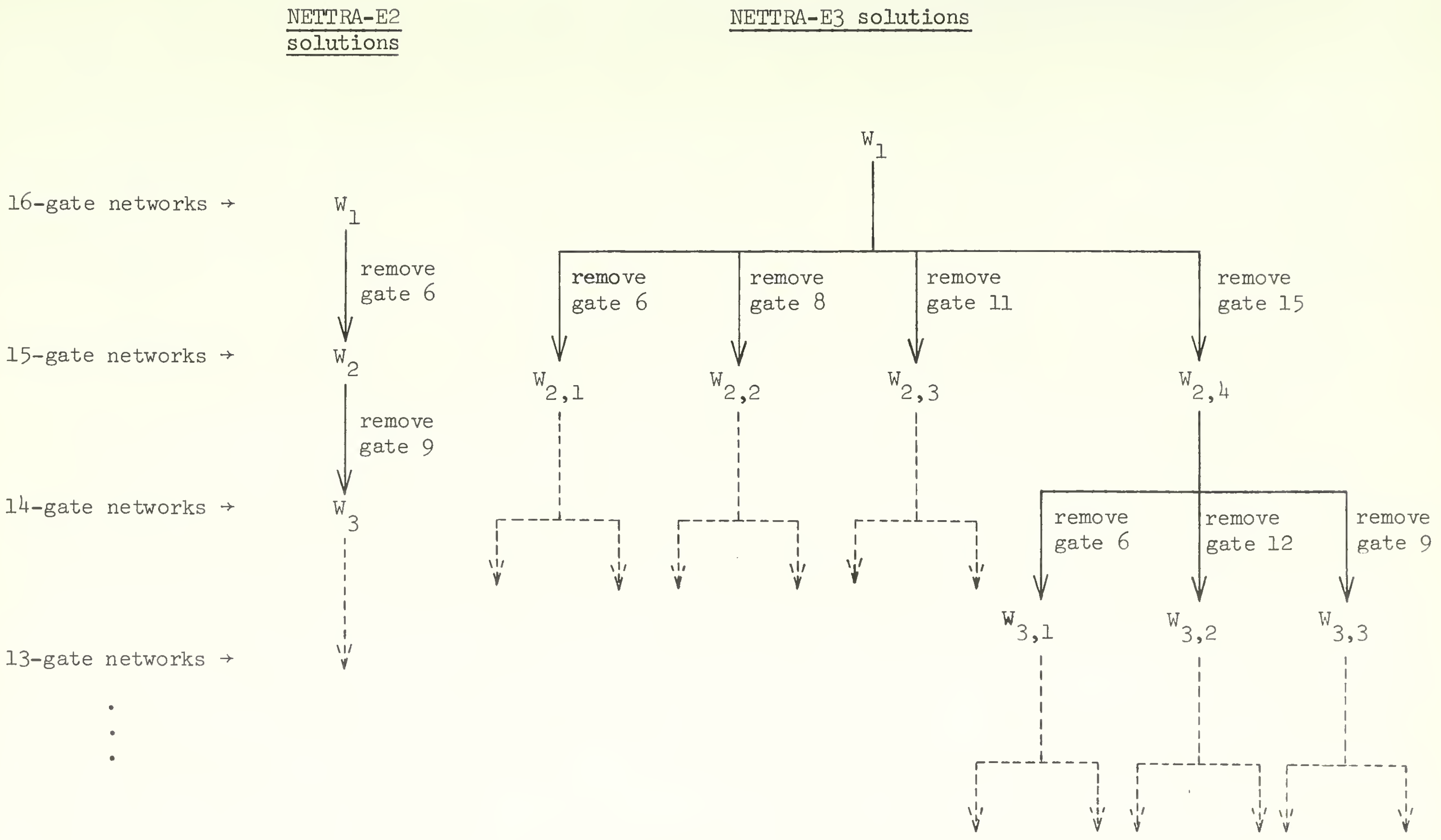


Figure 3.2.1 Typical intermediate solutions produced by NETTRA-E2 and -E3.

While NETTRA-E3 will obviously produce a solution as good as the best solution produced by NETTRA-E2, it is quite possible that it might produce an even better solution. Of course NETTRA-E3 can require much more computation time than NETTRA-E2, so each program has an advantage over the other, depending on the intended application.

3.2.1 Program organization of NETTRA-E3

NETTRA-E3 needs more storage space than NETTRA-E1 or -E2 due mainly to the addition of the previously mentioned stack. The program requires 207K bytes of core storage, about 82K for the actual program instructions and about 125K for the stored data.

The following subroutines, written in FORTRAN IV for the IBM 360/75, constitute the program NETTRA-E3: ALPATH, CALS1, CONECT, FORC, MAIN (this differs from the MAIN in NETTRA-E1 or -E2), MINI2, ORDRQ2, OUTPUT, POT, PROCCE (this differs from the PROCCE found in both NETTRA-E1 and -E2), RCEC, RPLCF, and SUBNET. Two system-supplied timing subroutines, STIMEZ AND KTIMEZ are also assumed to be available, but if they are not, their use can be omitted from the program, or another suitable timing routine substituted, without harming the procedure itself.

Figure 3.2.1.1 illustrates the general organization of the program. In this figure, an arrow from block i to block j represents the fact that the subroutine in block i calls the subroutine in block j.

In NETTRA-E3, subroutine MAIN calls the subroutine ALPATH only once. After that, it is ALPATH that controls the application of the error-compensation procedure to the networks stored in the stack. The modified version of PROCCE stores the intermediate solutions (networks) in the stack in the order

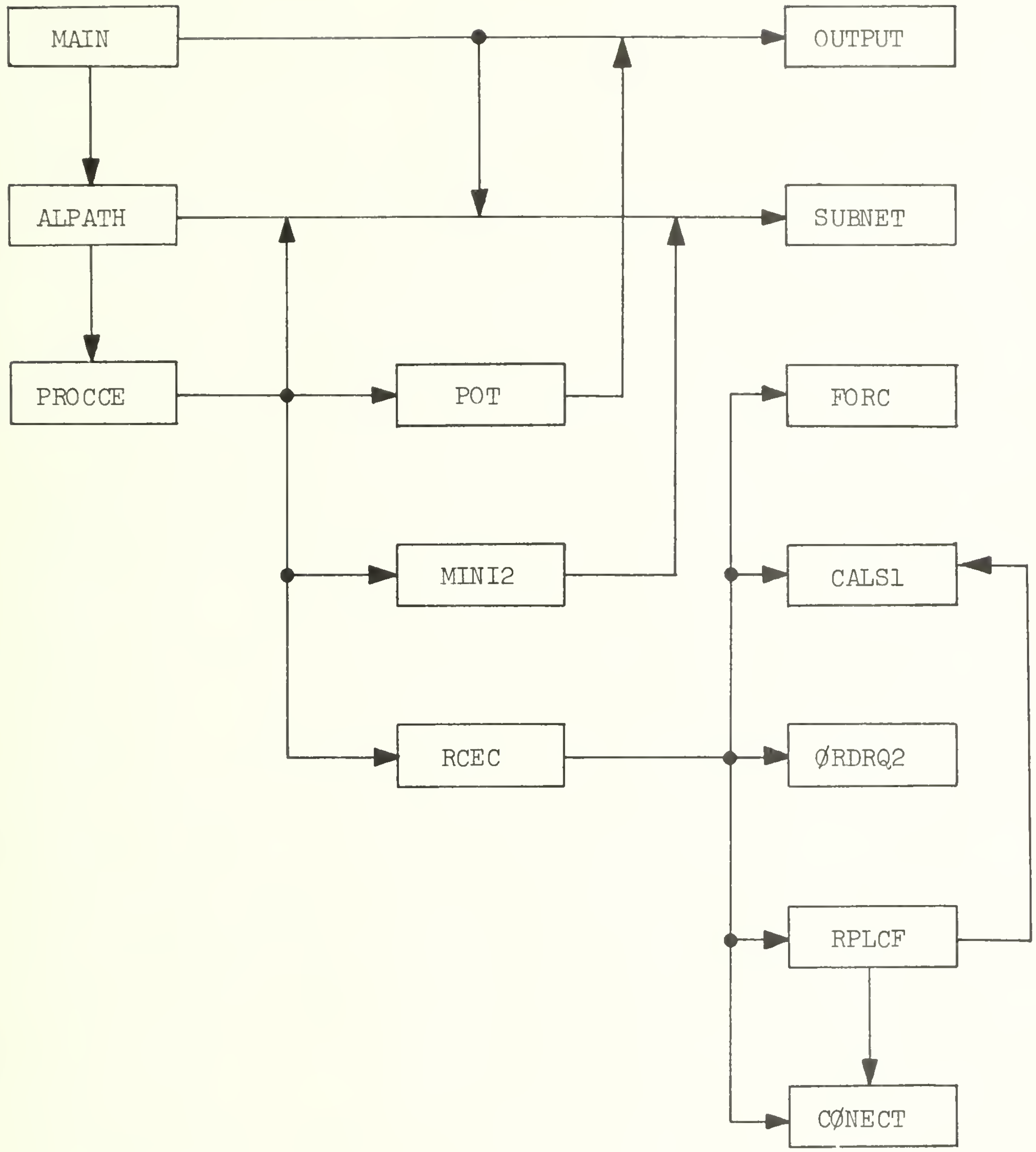


Figure 3.2.1.1 General organization of the program NETTRA-E3.

in which it obtains them. NETTRA-E3 terminates when the stack empties.

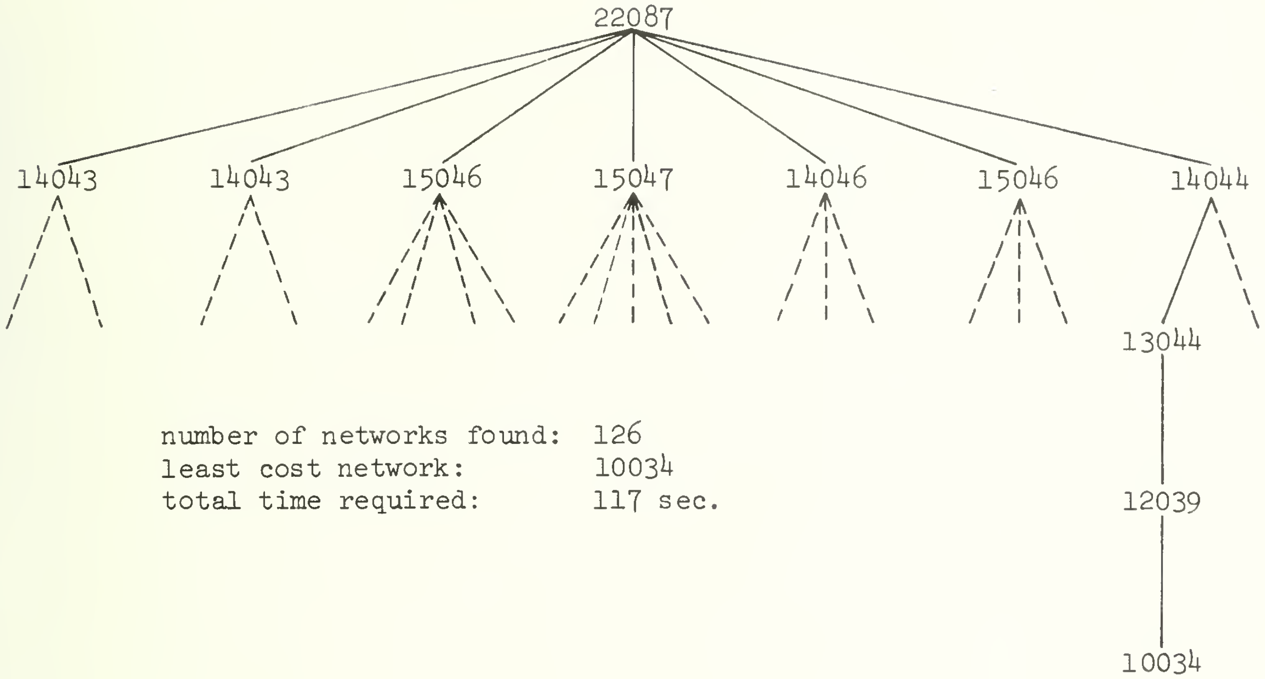
Once again, the input data setup is the same as that for NETTRA-E1 and -E2. The details can be found in Section 5. Listings of all of the subroutines used in NETTRA-E3 can be seen in the appendix.

3.2.2 Example of NETTRA-E3

As in the example for NETTRA-E2, this example also begins with the same initial network described in Figure 2.4.1 (a), (b), and (c). However, NETTRA-E3 produces a whole "tree" of 126 transformed, reduced networks (note that many of these can be and probably are identical) compared with the single "string" of three transformed, reduced networks derived by NETTRA-E2 from the same initial network. This tree is partially pictured in Figure 3.2.2.1 where the numbers represent the number of gates and connections in the corresponding network. (Each number is determined by multiplying the number of gates by 1000 and adding the number of connections. For example, 22087 represents a network of 22 gates and 87 connections.)

It can be seen that seven reduced networks are obtained from the initial network alone. And from each of these "sons" of the initial network, two or more additional networks are produced, etc. It requires 117 seconds of computation time to completely generate this tree of solutions (i.e., transformed, reduced networks).

Only one "path" of solutions is completely represented in Figure 3.2.2.1. It terminates with a network consisting of ten gates and 34 connections. This is the smallest size of any of the 126 transformed networks (although it is not the only network of such size in the tree). The printout obtained from NETTRA-E3 for this network is shown in Figure



number of networks found: 126
least cost network: 10034
total time required: 117 sec.

Figure 3.2.2.1 Solution tree found by NETTRA-E3.

```
          NETWORK NUMBER   32 DERIVED BY PROCCE.

          THE PARENT OF THIS NETWORK IS NUMBER   31

TRUTH TABLE

X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

X2 = 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

X3 = 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1

X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1

X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

 1 = 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0 1 0 1 1 1 0 0 1 1 1 0 1 0 0 0

 2 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 3 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 4 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 5 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 6 = 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0

 7 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 8 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 9 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

10 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

11 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

12 = 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0

13 = 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

14 = 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0

15 = 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0

16 = 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0

17 = 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0

18 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

19 = 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 1 1 1
```

(a) Truth table for transformed, reduced network.

Fig. 3.2.2.2 Printout of a transformed, reduced network obtained by NETTRA-E3.

```
GATE .. LEVEL      FED BY

   1     / 1/      10  12  19

   2     / 1/

   3     / 1/

   4     / 1/

   5     / 1/

   6     / 3/      X4  X5  14

   7     / 1/

   8     / 1/

   9     / 1/

  10     / 2/      X1  13

  11     / 1/

  12     / 2/      X2  X5   6  17

  13     / 5/      X1  X5

  14     / 4/      X2  X3  13

  15     / 3/      X2  X4

  16     / 3/      X3  X4

  17     / 3/      X3  X5  14

  18     / 1/

  19     / 2/       6  13  14  15  16  17

    THIS NETWORK HAS A COST OF: 10030
```

(b) Configuration of final network.

3.2.2.2. A similar printout is made for each of the 126 networks found by NETTRA-E3.

This particular network is assigned the name "Network #32". It is derived from "Network #31" which is the network corresponding to the network of cost 12039 appearing in Figure 3.2.2.1. The truth table for Network #32 appears in Figure 3.2.2.2 (a), and the network's configuration description is in Figure 3.2.2.2 (b).

Thus, for the particular initial network in Figure 2.4.1 (c), NETTRA-E3 is able to obtain a better solution (i.e., a solution of less cost) than that obtained by NETTRA-E2 (see Figure 3.1.2.1).

4. MAJOR FUNCTIONS OF SUBROUTINES

Thirteen different subroutines are used in the programs described in this manual: ALPATH, CALS1, CONECT, FORC, MAIN, MINI2, ORDRQ2, OUTPUT, POT, PROCCE, RCEC, RPLCF, and SUBNET. Complete program listings of these subroutines (a couple of subroutines have more than one listing since they appear in slightly different forms in different programs) can be found in the appendix.

The main functions of these thirteen subroutines are as follows:

ALPATH: This subroutine is used only in NETTRA-E3. It controls the calling of subroutine PROCCE for the various networks produced in the multi-path mode which are stored in a stack. Instead of the sequence of solutions (networks) of decreasing cost produced by NETTRA-E2, this subroutine causes the production of a "tree" of solutions.

CALS1: Subroutine RCEC calls subroutine CALS1 in several places. When CALS1 is entered, the following sets of functions are given: (1) set S which is a set of to-be-replaced input functions of the gate under considerations, (2) set S2 which contains candidate functions for the replacement of functions in set S. CALS1, based on these two sets, calculates a subset S1 of S consisting of the functions that may be replaced by functions in set S2. Since set S2 may contain mutually incompatible functions, it is possible that set S2 cannot actually replace the calculated set S1. In such a case, CALS1 will be reentered from subroutine RPLCF with a new restricted S2.

CONECT: This subroutine has one argument which is to specify a function in the possible output table. When this subroutine is called from RCEC, it actually constructs (make necessary connections) this function and connect it to the gate under consideration.

FORC: This subroutine is called by RCEC before other error-compensation procedures are applied. It removes redundant connections among input connections of the gate under consideration.

MAIN: This subroutine repeatedly reads in groups of input data which include information about the given networks, such as the number of external variables, the availability of the complements of variables as input variables, the number of output functions, the number of NOR gates, the list of connections, and the truth table of the output functions (see Section 5 for details). Using this information, MAIN constructs the incidence matrix, INC$MX, for the network. INC$MX is a two-dimensional array whose arguments represent gates or external variables. An array element INC$MX(GI, GJ) $\geq$ 1 indicates a connection from GI to GJ; an array element INC$MX(GI, GJ) $\leq$ 0 indicates the absence of a connection from GI to GJ. Next, subroutine SUBNET is called to calculate the level of each gate and to make lists of predecessors and successors (i.e., which gates precede which and which gates succeed which). MAIN then prints out the truth table and the constructed incidence matrix of the original network by calling the subroutine OUTPUT. Finally the desired transduction procedure is applied to the network by calling the subroutine(s) realizing that procedure. The transformed, reduced network is stored in INC$MX, replacing the original network. Then MAIN prints the results of the transduction procedure, i.e., the new incidence matrix and the new truth table.

<u>MINI2</u>: Subroutine PROCCE, when initially entered, calls MINI2 to eliminate quickly some easily removable gates in the given network. MINI2 is a subroutine which realizes a pruning procedure (i.e., it transforms a network strictly by removing connections), and it is described in some detail in [7]. Calling the entry point FORMGO simply forms an ordering of the gates of the network and stores it in the array GORDER for future use. A call to INITGS, another entry point of MINI2, initializes the CSPF vectors of the gates of a network in preparation for calling RCEC to compensate for errors in that network.

<u>ORDRQ2</u>: This subroutine arranges the input connections of the selected gate according to an ordering based on the number of components covering 0-error-components. Two arrays are used to store two groups of input connections from gates. Group 1 contains functions which have at least one primary 1-error-component, whereas group 2 contains functions which have 1-error-components other than primary ones. Both arrays are sorted according to the number of 1-error-components in each function.

<u>OUTPUT</u>: This subroutine may be entered at five different points by a call

to either OUTPUT, PAGE, LINE, TRUTH or CKT.

OUTPUT	assigns mnemonic names to external variables and gates for the purpose of achieving a readable printout.
PAGE	ejects one page on the printer.
LINE	skips a specified number of lines on the printout sheet. The number is specified by the argument in the call (e.g., "CALL LINE(5)" skips 5 lines).
TRUTH	prints out the truth table of the network currently stored in INC$MX.
CKT	prints out the network itself.

POT: This is the subroutine which constructs the potential output table. The procedure and the representation of the table are discussed in great detail in Section 2.2.2.

PROCCE: This subroutine directs the execution of the error-compensation procedure. It does not itself perform the detailed logic of the procedure, but it controls the sequence of calling the subroutines which do (primarily, RCEC, POT, and MINI2).

RCEC: This procedure realizes the essential part of the entire error-compensation procedure. It has two returns: RETURN1 will be executed if no error-compensation can be performed, whereas RETURN2 will be executed if some error-compensations are performed. In the later case, PROCCE will compare the outputs of the new network with the specified outputs. Depending on the result of comparison, RCEC will be reentered unless a network realizing the given functions has been obtained.

RPLCF: This subroutine is called by RCEC immediately after it calls CALS1. From set S1 calculated by subroutine CALS1 and set S2, it calculates a subset S3 of S2 which is necessary for replacing set S1. Because of the incompatibility of functions in S2, this subset may not actually exist. In this case, some incompatible functions in set S2 will be prohibited and a new set S1 will be calculated by calling CALS1. RPLCF will then repeat the procedure from the beginning.

SUBNET: This subroutine may be entered at any of three different points by a call to either SUBNET, UNNECE, or PVALUE.

SUBNET generates detailed information on the network configuration stored in INC$MX: (1) it calculates the level of each gate in the network. Level 1 is assigned to gates having no output connections (thus all gates which have been removed from the network will be assigned level 1). (2) It lists all immediate successors and immediate predecessors for each gate. (3) It calculates the successor matrix which is stored in a two-dimensional array, SUC$MX. The value of SUC$MX(GI, GJ) indicates the existence or non-existence of a path from gate (or external variable) GI to gate GJ.

UNNECE disconnects certain types of obviously unnecessary connections in the network and updates the above information (discussed in (1), (2), and (3)). The connections removed from the given network are those existing in no paths from the external variables to the output gates.

PVALUE calculates the actual truth table for the entire network stored in INC$MX.

5. INPUT DATA SETUP

In order to fully understand the description of the setup of the input data cards, certain preliminary explanations are necessary.

The purpose of network transductions is to reduce the cost of a network which realizes a certain function (or functions) or to alter the network in such a way as to allow another transduction to eventually accomplish such a reduction. This cost, C, is formally defined by the weighted sum of the number of gates, R, and the number of connections[†], I, of a particular network, i.e.,

$$C = A \times R + B \times I$$

where weights A and B are arbitrary non-negative numbers.

Suppose the original network which is to be transformed produces m output functions of n variables. Let x_ℓ, $\ell = 1, \ldots, n$, be the external variables and f_h, $h = 1, \ldots, m$, be the output functions. Before a transformation can be performed on a network by a program, a description of that network must be input to the program. In the case when all of the output functions are completely specified (i.e., no "don't cares"), specifying only the interconnection pattern of the network is sufficient. But if one or more of the output functions is not completely specified, then the user must also provide the truth table (truth tables for all output functions are condensed into a single table) of the problem. Providing the truth table to the program consists of two steps,

[†] A "connection" refers to either a connection from an external variable or an interconnection between two gates.

namely the specification of external variables, and the specification of output functions.

The method of specifying the output functions depends directly upon the method chosen to specify the external variables. External variables may be specified in either of two ways, (a) an implicit specification of external variables, or (b) an explicit specification of external variables.

(a) In the case of implicit specification of external variables, the user specifies the number n of external variables along with a parameter which indicates whether or not the uncomplemented variables are available. Reading the value n along with the parameter, the program internally generates the entries of external variables of an ordinary truth table, that is, a truth table which consists of 2^n input vectors, as shown in Fig. 5.1. In this truth table, the input vectors are arranged according to the order such that an integer j expressed in a binary representation $(x_1 \ldots x_n)$ increases, where x_1 is the most significant digit and x_n is the least significant digit. For example, the truth table for a function of three variables is shown in Fig. 5.2.

The implicit specification of external variables is used for logical design problems in which the output functions have relatively few don't-care terms.

The uncomplemented variables	x_1	$x_1^0 \dots x_1^j \dots x_1^{2^n -1}$	
	$\cdot$	$\cdot$	
	$\cdot$	$\cdot$	
	$\cdot$	$\cdot$	
	x_n	$x_n^0 \dots x_n^j \dots x_n^{2^n-1}$	
The complemented variables	$\bar{x}_1$	$\bar{x}_1^0 \dots \bar{x}_1^j \dots \bar{x}_1^{2^n-1}$	These entries exist only in the case of logical design problems where the complemented variables are available as external inputs.
	$\cdot$	$\cdot$	
	$\cdot$	$\cdot$	
	$\cdot$	$\cdot$	
	$\bar{x}_n$	$\bar{x}_n^0 \dots \bar{x}_n^j \dots \bar{x}_1^{2^n-1}$	
The output functions	f_1	$f_1^0 \dots f_1^j \dots f_1^{2^n-1}$	
	f_2		
	$\cdot$	$\cdot$	
		$\cdot$	
	$\cdot$	$\cdot$	
	f_m	$f_m^0 \dots f_m^j \dots f_m^{2^n-1}$	

Fig. 5.1 The truth table of output functions of n variables

x_1	0	0	0	0	1	1	1	1	
x_2	0	0	1	1	0	0	1	1	
x_3	0	1	0	1	0	1	0	1	
$\bar{x}_1$	1	1	1	1	0	0	0	0	These entries exist only in the case of logical design problems where complemented variables are available as input variables.
$\bar{x}_2$	1	1	0	0	1	1	0	0	
$\bar{x}_3$	1	0	1	0	1	0	1	0	
f_1	f_1^0	.	.	.	.	.	.	f_1^7	

Fig. 5.2 The truth table of a function of three variables.

(b) In the case of explicit specification of external variables, the user specifies the entries of external variables of the truth table using additional cards called < external-variable-card > s. The explicit specification of external variables is used in the case of logical design problems where output functions have many don't-care terms. Suppose that the output functions are defined for a subset of input vectors of the entire truth table of Fig. 5.1. Let $\vec{x}^{\,j}$, $j = j_1, j_2, \ldots, j_\mu$ denote these input vectors. The user can 'condense' the truth table of Fig. 5.1 into another table shown in Fig. 5.3.

$$
\begin{array}{ll|llll}
 & & \multicolumn{4}{c}{\overbrace{\hspace{6cm}}^{\text{only } \mu \text{ input vectors}}} \\
\text{The uncomplemented} & x_1 & x_1^{j_1} & x_1^{j_2} & \ldots & x_1^{j_\mu} \\
\text{variables} & \vdots & \vdots & & & \\
 & x_n & x_n^{j_1} & x_n^{j_2} & \ldots & x_n^{j_\mu} \\
\text{The complemented} & \bar{x}_1 & \bar{x}_1^{j_1} & \bar{x}_1^{j_2} & \ldots & \bar{x}_1^{j_\mu} \\
\text{variables} & \vdots & \vdots & & & \\
 & \bar{x}_n & \bar{x}_n^{j_1} & \bar{x}_n^{j_2} & \ldots & \bar{x}_n^{j_\mu} \\
\hline
 & f_1 & f_1^{j_1} & f_1^{j_2} & \ldots & f_1^{j_\mu} \\
 & \vdots & \vdots & & & \\
 & f_m & f_m^{j_1} & f_m^{j_2} & \ldots & f_m^{j_\mu}
\end{array}
$$

These entries exist only in the case of logical design problems where the complemented variables are avaiable as external inputs.

Fig. 5.3 A 'condensed' truth table having only the input vectors $\vec{x}^{j}$, $j = j_1, \ldots, j_\mu$, for which the output functions are defined.

Using < external-variable-card > s, the user can set up internally the table of Fig. 5.3 in place of Fig. 5.1.

5.1 Input Data Card Format

For each separate problem, a set of input data cards must be submitted which consists of the following[†]:

(i) < heading-card >
(ii) < problem-parameter-card >
(iii) < external-variable-card > s
(iv) < output-function-card > s
(v) < connection-description-card > s

(iii) and (iv): omitted for certain cases

Both (i) and (ii) will always consist of only a single card, but (iii), (iv), and (v) may each consist of several cards. Furthermore, types (iii) and (iv) are omitted if all output functions are completely specified, and (iii) need only be prepared in the case of the explicit specification of external variables for the truth table. Following is a description of the formats for each type of input card, (i), (ii), (iii), (iv) and (v):

(i) < Heading-card >

This is the first card of the input deck for a problem. This card may contain any alphanumeric information, in columns 1 ~ 80, which may be used for the identification of the problem, but none of the information on this card will be used in the actual computation. This information will be printed on the first page of the output.

[†] The current implementations of the NETTRA programs accept only heading, problem-parameter, and connection-description cards. Eventually it is hoped that these programs will be modified to accept all of the options described in this section.

(ii) < Problem-parameter-card >

This card specifies the nature of the problem the user wants to solve. There are 7 fields in which to specify the parameters with characters and numerals. These fields are as follows:

Cols. 1 ~ 4: An integer, N, which is right-justified.

This number, N, represents the number of external variables, n, of the output functions. Be sure to punch n (rather than 2n) for N in the case of both complemented and uncomplemented variables available.

Cols. 5 ~ 8: An integer, M, which is right-justified.

This number, M, is the number of output functions, m, to be realized simultaneously. Therefore, of course, M will also be the number of output gates in the network.

Cols. 9 ~ 12: An integer, R, which is right-justified.

This number, R, specifies the number of gates which are included in the network. For various reasons, the user may wish to input networks in which one or more of the gates are "isolated" (i.e., are not connected to any other gates). This is permissible as long as these "isolated" gates are also included in the total number of gates, R.

Cols. 13 ~ 16: An integer, A , which is right-justified.

The number A is the value of the non-negative weight for the number of gates in the cost function. (See Table 5.1.1, 'Typical combinations of values A and B for different network reduction problems'.)

Cols. 17 ~ 20: An integer, B , which is right-justified.

The number B is the value of the non-negative weight for the number of connections in the cost function. (See Table 5.1.1.)

Col. 24: A blank 'b'[†], or one of the characters, 'C', 'X', 'Y', 'U' or 'V'.

The 'b' or 'C' parameter represents an implicit specification of both the external variables and an implicit specification of the output functions (in this case, the output functions will be calculated from the connection pattern of the network). The 'X' or 'Y' parameter indicates an implicit specification of external variables only. The 'U' or 'V' parameter indicates an explicit specification of external variables. (See summary of these symbols in Table 5.1.2)

The 'b' or 'X' parameter specifies that only uncomplemented external variables are available for the network. The 'C' or 'Y' parameter specifies that both uncomplemented and complemented variables are available for the network. If the user specifies the 'b', 'X', 'C', or 'Y' parameter, the program sets up the truth table by generating a set of 2^n input vectors $(x_1^j, \ldots, x_n^j)$, for $j = 0, \ldots, 2^n - 1$, in the case of a 'b' or 'X' parameter, or $(x_1^j, \ldots, x_n^j, \bar{x}_1^j, \ldots, \bar{x}_n^j)$ for $j = 0, \ldots, 2^n - 1$, in the case of a 'C' or 'Y' parameter.

The 'b' or 'C' parameters should be used for problems in which the output functions contain no don't-care terms. For such problems, the preparation of the < external-variable-card > s and the < output-function-card > s can be dispensed with since the program can calculate completely all output functions using only a description of the

[†] A 'b' stands for a blank (i.e., no character punched).

Network Reduction Problem	Values of A and B
reducing the number of gates only.	$A = 1$ and $B = 0$
reducing the number of gates primarily, then reducing the number of connections secondarily.†	$A = 100$ and $B = 1$
reducing the number of connections only.	$A = 0$ and $B = 1$
reducing the number of connections primarily, then reducing the number of gates secondarily.	$A = 1$ and $B = 100$
reducing the sum of the number of gates and the number of connections.	$A = B = 1$

Table 5.1.1 Typical combinations of values A and B for different network reduction problems.

† Most of the programs in the NETTRA system are oriented toward this reduction problem, so the user will probably find this combination of A and B the most useful.

uncomplemented variables only available	both complemented and uncomplemented variables available	
'b'	'C'	implicit specification of external variables and output functions
'X'	'Y'	implicit specification of external variables
'U'	'V'	explicit specification of external variables

Table 5.1.2 Possible symbols for column 24 of < problem-parameter-card >.

connection pattern of the network (provided by the <connection-description-card>s).

Similarly, the 'X' or 'Y' parameter implies the use of a complete truth table (i.e., 2^n input vectors for n external variables) inside the program. Since from this information the program can easily generate the truth table entries for the external variables, as just mentioned, the < external-variable-card > s are unnecessary. The m < output-function-card > s, however, must still be prepared.

The 'U' parameter specifies that only uncomplemented external variables are available for the network. The 'V' parameter specifies that both uncomplemented and complemented variables are available for the network. In either case, the 'U' or the 'V' parameter, the user must prepare n < external-variable-card > s and m < output-function-card > s. The program sets up the truth table by reading these < external-variable-card > s and < output-function-card > s.

Cols. 25 ~ 28: An integer, NEPMAX, which is right-justified.

This parameter is omitted for all NETTRA programs except those involving "error-compensation" routines. In the cases where NEPMAX is required, a further discussion of this parameter can be found elsewhere in the manual. The abbreviation NEPMAX is a mnemonic for "maximum number of error positions", and the default is $\text{NEPMAX} = 2^{(n-1)}$, where n is the number of external variables.

(iii) < External-variable-card > s

In combination with the 'U' or 'V' parameter in column 24 of the < problem-parameter-card >, the n < external-variable-card > s specify the entries of external variables of the truth table of

Fig. 5.3. Each card contains the binary representation of external variable x_ℓ, i.e., $(x_\ell^{j1}, x_\ell^{j2}, \ldots, x_\ell^{j\mu})$, starting from column 1 of the card. The maximum number of bits in a binary representation is limited to 32. (This means the maximum number of input vectors is 32.) If the actual number of bits is less than 32, then a termination symbol '/' (slash) is put on the right of the right-most bit of the binary representation on the first < external-variable-card >. The remaining columns after the termination symbol '/' in the first card, as well as the same columns in the following cards, may contain any alphanumeric information which may be used for identification. This information will not be printed on the output pages.

In the case of the 'V' parameter, the program generates the binary representations corresponding to complemented variables by taking negations of the entries of the < external-variable-card > s. Therefore the user must not provide < external-variable-card > s representing the complemented variables, $\bar{x}_\ell$.

If one of the parameters 'b', 'C', 'X', or 'Y' appears in column 24 of the < problem-parameter-card >, the user does not prepare < external- variable-card > s.

(iv) < Output-function-card > s

The m < output-function-card > s specify the set of m output functions to be realized simultaneously. Each card contains the binary representation of one output function f_h, starting from column 1 of the card. A symbol '*' is used to denote don't-care terms, if any. The maximum number of bits in a binary representation is limited to 32.

The actual number of bits must be 2^n in the case of an implicit specification of external variables, or must be the same as defined by the < external-variable-card > s in the case of an explicit specification of external variables. The remaining columns, up to column 72 (inclusive), may contain any alphanumeric information which may be used for identification. This information will not be printed on the output pages.

If either the 'b' or 'C' parameter appears in column 24 of the < problem-parameter-card >, the < output-function-card > s must be omitted.

(v). < Connection-description-card > s

In the present version of the program, 9 cards (some of which may be just blank cards) are required.† Each of these cards is divided into 16 fields of 5 columns each (i.e., columns 1 ~ 5, 6 ~ 10, 11 ~ 15, ..., 71 ~ 75, 76 ~ 80). Beginning with the first field of the first card, continuing through the succeeding fields of that card and through the fields of as many additional cards as necessary (up to a maximum of 9, total), the expressions (explained in the next paragraph) C_1, C_2, C_3, ..., are punched right-justified in their respective fields.

Each gate of the network is labeled uniquely by assigning it one of the integers 1, 2, ..., R, such that the output gates receive

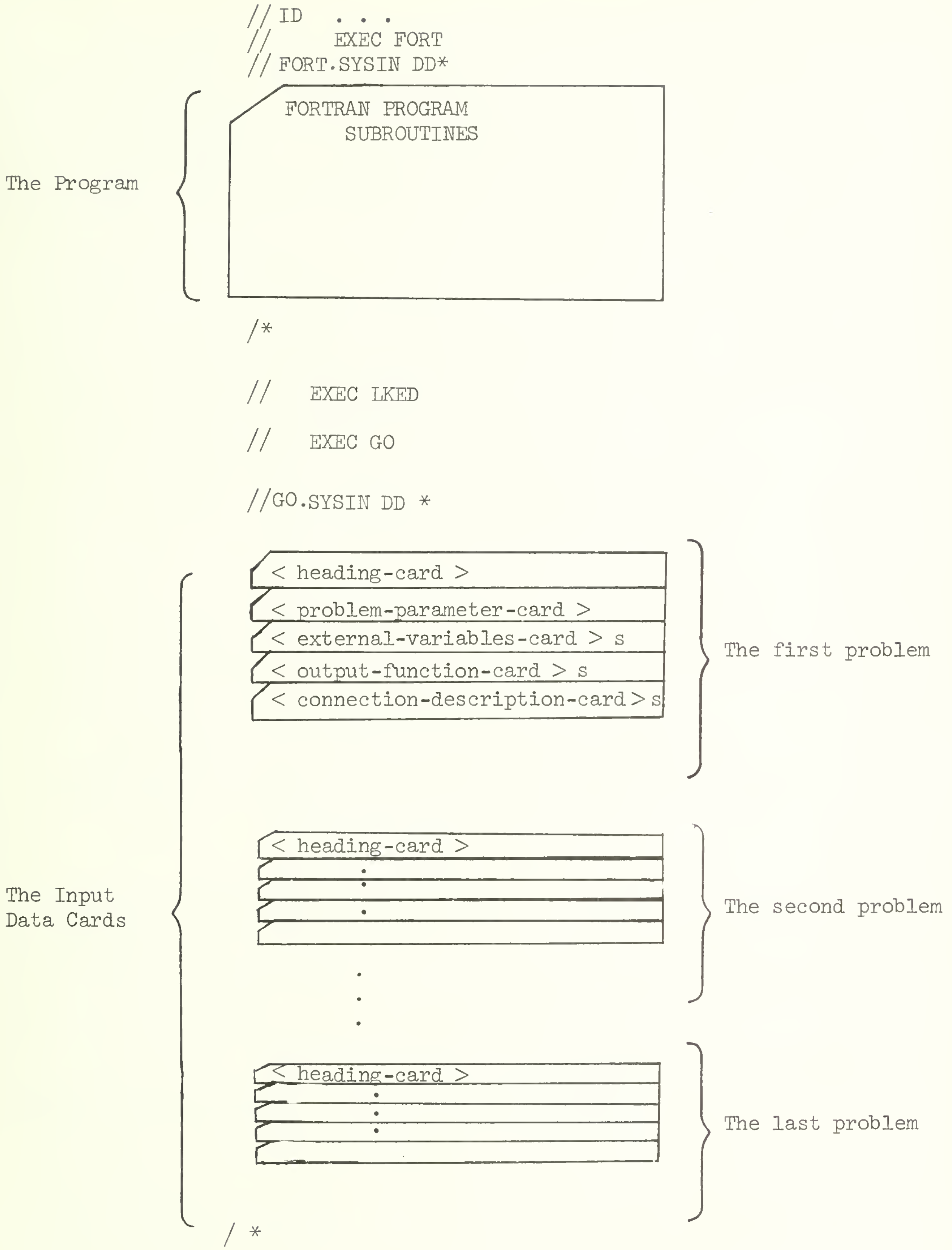
† For many uses, the user will probably find that these 9 cards far exceed his needs, and may thus be inconvenient. In such a case, the number of required cards may be easily adjusted by making the obvious changes in two statements (A READ statement and a DO statement) following the comment card "C**** READ IN NETWORK INFORMATION AND SET UP INC$MX *****" in subroutine MAIN.

the labels 1, 2, ..., m. The names X1, X2, ..., Xn are assigned to the external variables x_1, x_2, ..., x_n (and the names Y1, Y2, ..., Yn to the complemented external variables $\overline{x}_1$, $\overline{x}_2$, ..., $\overline{x}_n$, if appropriate).[†] Now, for each connection of the network (i.e., including both the connections from external variables to gates and connections from gates to other gates), a 4 character expression, C_i, is formed, to represent that connection as follows: to represent a connection from gate GI to gate GJ, the numeric label GI is inserted into the first two character positions of C_i and the numeric label GJ is inserted into the second two positions (e.g., the C_i for a connection from gate 9 to gate 5 would be "0905"); to represent a connection from external variable XI to gate GJ, the alphanumeric label XI is inserted into the first two character positions of C_i and the numeric label GJ into the second two positions (e.g., the C_i for a connection from external variable x_3 to gate 10 would be "X310").

Every connection of the network must be represented by a C_i, although there are no restrictions on the order in which the connections (i.e., C_i's) are punched onto the input cards.

[†] At the time of writing, the programs have not yet been changed to recognize this new labeling system. The old labels are simply: 1, 2, ..., n, for external variables x_1, x_2, ..., x_n (and $n+1$, $n+2$, ..., $2n$ for the complemented variables $\overline{x}_1$, $\overline{x}_2$, ..., $\overline{x}_n$, if they are permitted in the problem); $n+1$, $n+2$, ..., $n+m$ for the m output gates of the network ($2n+1$, $2n+2$, ..., $2n+m$ if complemented variables are included); and finally $n+m+1$, $n+m+2$, ..., $n+R$ ($2n+m+1$, $2n+m+2$, ..., $2n+R$) for the non-output gates of the network.

These five groups of cards, (i), (ii), (iii), (iv) and (v) in the correct order constitute the necessary description for a single problem. In order to solve several problems during the same computer run, the descriptions for the desired problems are just arranged serially. See Fig. 5.1.1 for an example of the sequencing of all cards for the execution of a NETTRA program using typical JCL statements for the IBM 360/75.



```
//ID  . . .
//     EXEC FORT
//FORT.SYSIN DD*
    FORTRAN PROGRAM
         SUBROUTINES
/*

//   EXEC LKED

//   EXEC GO

//GO.SYSIN DD *
< heading-card >
< problem-parameter-card >
< external-variables-card > s
< output-function-card > s
< connection-description-card > s

< heading-card >
      .
      .
      .

      .
      .
      .

< heading-card >
      .
      .
      .
/ *
```

Fig. 5.1.1 Input card sequence for the execution of a typical NETTRA program.

5.2 Restrictions on Problem Size

In order to fit the programs into a finite amount of space, some restrictions on the size of an acceptable problem are required:

1. The number t of input vectors in the truth table is 32 or less.
2. The number n of external variables.

   Because of $t \leq 32$, n is 5 or less in the case of completely specified functions. In the case of incompletely specified functions, however, any $n \leq 20$ is acceptable if only uncomplemented variables are available, or $n \leq 10$ if both uncomplemented and complemented variables are available, provided that the truth table is defined by the < external-variable-card > s.
3. The number R of gates.

   The number of gates, R, may not exceed 40-n in the case of only uncomplemented variables available (a 'b', 'X', or 'U' parameter). In the case of both uncomplemented and complemented variables available (a 'C', 'Y' or 'V' parameter), the limit is lowered to 40-2n.

All of these limitations are essentially imposed by the array sizes in the programs as presently written. To loosen the restrictions is mainly a task of increasing array dimensions appropriately.

5.3 Examples of Input Data Setup

The following examples will illustrate, for the general program in the NETTRA system, various possible input data card setups complying with the directions given in Section 5.1.

Example 1: A two output network of four variables shown in Fig. 5.3.1. Assume the two output functions are $f_1 = \text{CCEF}$[†] and $f_2 = \text{3BBB}$ and only uncomplemented variables are available. Furthermore, assume it is desired to reduce the number of gates primarily and the number of connections secondarily (see Table 5.1.1).[††]

From this description, the < problem-parameter-card > must contain the following values:

Cols. 1 ~ 4	4,	the number of external variables
Cols. 5 ~ 8	2,	the number of output functions
Cols. 9 ~ 12	8,	the number of gates in the original network
Cols. 13 ~ 16	100,	the value of A
Cols. 17 ~ 20	1,	the value of B
Cols. 24	'b',	uncomplemented variables only available and implicit specification of both the external variables and the output functions
Cols. 25 ~ 28	'b',	since the NEPMAX parameter is unrelated to the program to be used

Fig. 5.3.2 shows the setup of data cards used to specify the network in Fig. 5.3.1 as input for the program. Notice that in forming the C_i, the four uncomplemented variables are represented by the labels X1, X2, X3, X4; the two output gates by the numbers 1, 2; and the remaining gates, by the numbers 3, 4, 5, 6, 7, 8. This manner of labeling is

[†] For convenience, the output functions are expressed in hexadecimal notation. When the numbers in this notation are expanded into binary, they represent the output vectors as they appear (i.e., in the same left-to-right order) in the complete truth table described earlier and pictured in Fig. 5.1.

[††] This assumption is implicit in most of the transduction procedures and their implementations which comprise the NETTRA system.

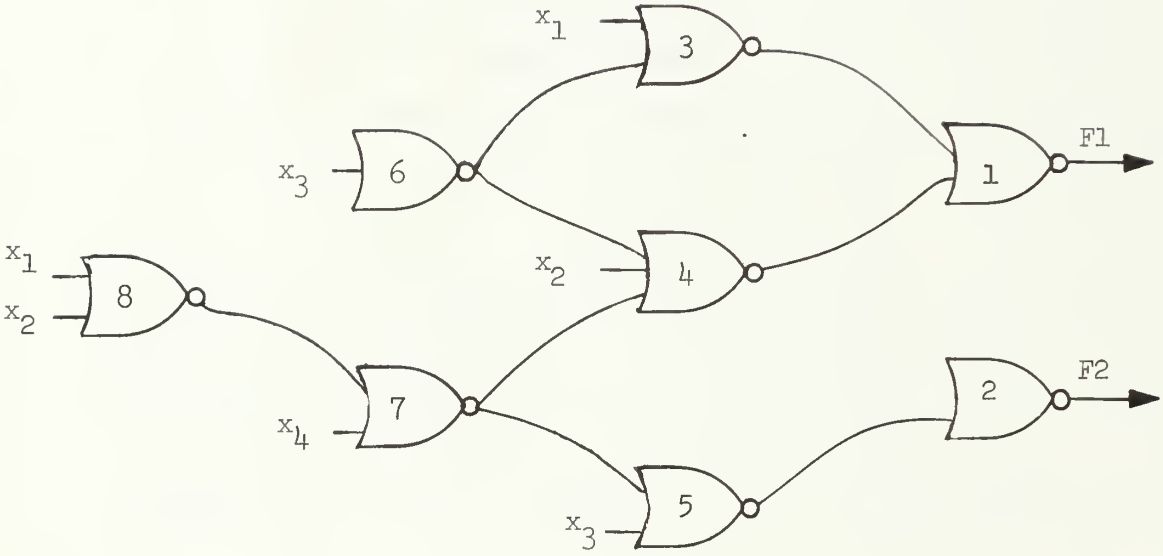


Fig. 5.3.1 Network to be transformed in Examples 1 and 2.

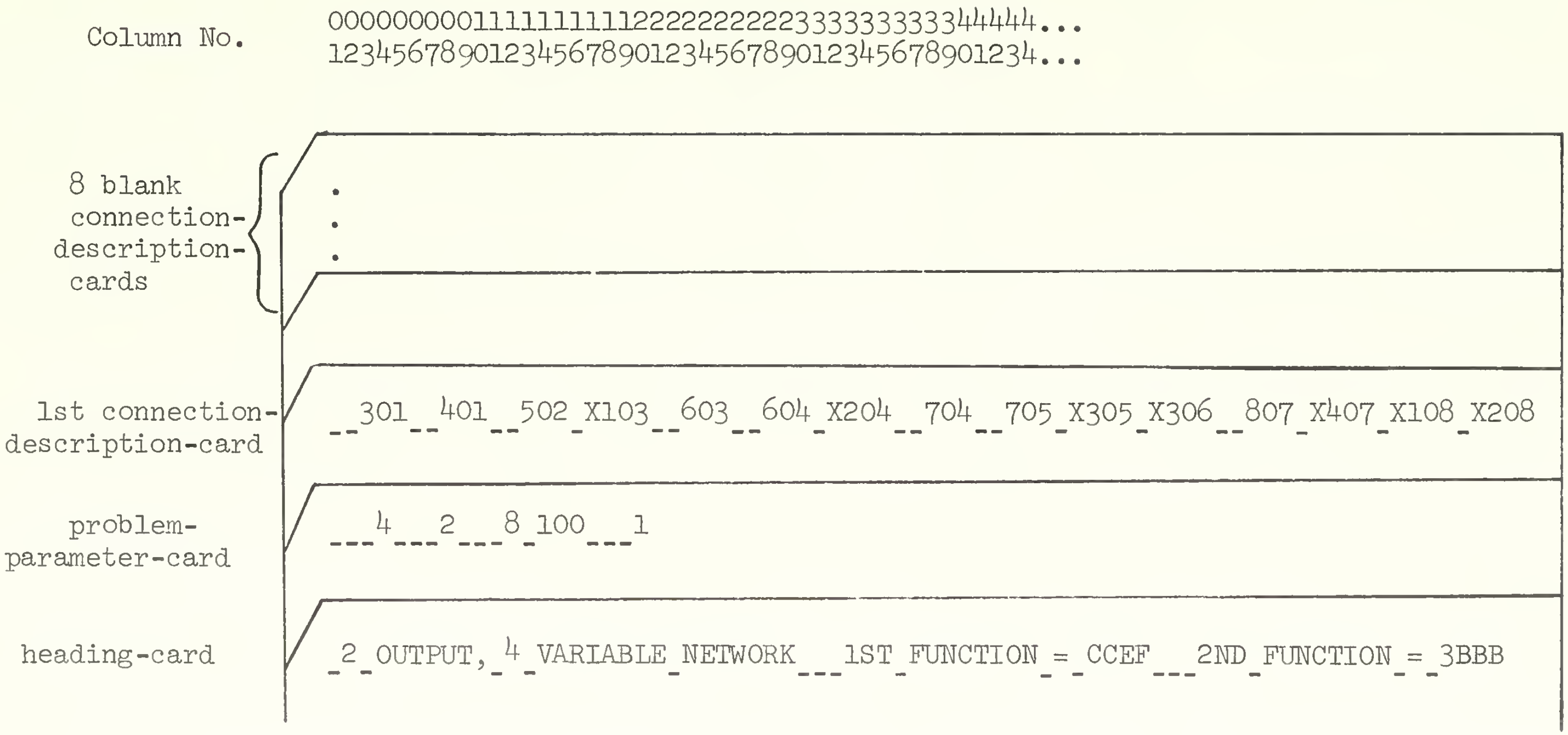

Column No.
000000000111111111122222222223333333333444444...
1234567890123456789012345678901234567890123 4...
8 blank connection-description-cards
.
.
.
1st connection-description-card
__301__401__502_X103__603__604_X204__704__705_X305_X306__807_X407_X108_X208
problem-parameter-card
___4___2___8_100___1
heading-card
_2_OUTPUT,_4_VARIABLE_NETWORK___1ST_FUNCTION_=_CCEF___2ND_FUNCTION_=_3BBB

Fig. 5.3.2 Possible setup of data cards to specify the problem given in Example 1.

strictly required by the instructions for preparing the < connection-description-card > s (see Section 5.1).

The heading card in Fig. 5.3.2 will simply be read by the program and printed character for character onto the output page as an identification of the particular problem. Below that, the number of variables, number of functions, and the cost coefficients, A and B, will be printed (all with appropriate labels). Also, immediately following will be a statement of what types of external variables are permitted (i.e., either just uncomplemented variables or both complemented and uncomplemented) along with their generic names:

$$\left.\begin{array}{l}\text{X - for uncomplemented variables}\\ \text{Y - for complemented variables}\end{array}\right\}\begin{array}{l}\text{if external variables}\\ \text{were implicitly specified}\end{array}$$

<u>or</u>

$$\left.\begin{array}{l}\text{U - for uncomplemented variables}\\ \text{V - for complemented variables}\end{array}\right\}\begin{array}{l}\text{if external variables}\\ \text{were explicitly specified}\end{array}$$

For example, if both X and Y appear as generic names (as would occur in the case of an implicit specification of external variables with both complemented and uncomplemented variables available) then the external variable names which appear on subsequent output pages will be X1, X2, ..., Xn and Y1, Y2, ..., Yn. Or, if both U and V appear as generic names (as would occur in the case of an explicit specification of external variables with both complemented and uncomplemented variables available) the external variable names which appear in the output will be U1, U2, ..., Un (for the uncomplemented variables) and V1, V2, ..., Vn (for the complemented variables). It should be noted, however, that the letters U and V, as used as replacements for X and Y (respectively) in the naming of external variables (e.g. U1, V1 instead of X1, Y1), appear strictly on the output pages of the program - they are not used internally in the program and they must not appear in the variable names punched on the < connection-description-card > s by the user. They are intended only as an aid to the user so that, at a glance at the transformed network in the output, he can easily distinguish whether the external variables were implicitly or explicitly specified for that particular problem.

Following the statement of whether only uncomplemented or both complemented and uncomplemented external variables are employed, the user will find next on the output page the cost of the original network which was input to the program. This is the cost which was defined in the beginning of Section 5.

The cost will be followed by a truth table (generally in the same form as Fig. 5.1) showing the outputs (0 or 1) of all of the gates in the network for every external variable input combination (i.e., combinations of 0's and 1's) of interest.

Finally, below the truth table will be printed a description of the network submitted as input. This is for documentation purposes, and it is also much more readable than the network description which appeared on the < connection-description-card > s. In this description, each gate is listed along with the names of the gates and external variables which feed it. Also, to assist the user in sketching the network from its description, the level of each gate in the network is included (gates which do not feed other gates are assigned to level 1, all other gates are assigned level numbers such that each gate is in a level one higher than the highest level gate directly fed by it).

All of the information just described will be printed before the execution of the transduction actually begins. This will be followed, beginning at the top of a new output page, by the network(s) actually obtained as a result of the computation. First the complete truth table of the transformed network will be printed, followed by a network connection description of the form just described above. Finally, the cost of the new network will be calculated and printed.

In this example, it was assumed that there were no "don't-cares" in the output functions implicitly specified by the input, thus no < external-variable-card > s or < output-function-card > s were included. In the next example, however, < output-function-card > s will be required in order to specify some of the components of the output functions as "don't-cares".

Example 2: The two output network of four variables shown in Fig. 5.3.1. This is the same network used in Example 1, but this time the output functions are not assumed to be completely specified. Let f_1 = '11001**01*10*111' and f_2 = '0**110111*111011' be the required functions. Also, suppose that both complemented and uncomplemented variables are desired to be available during the transduction. Again the problem is to reduce the number of gates primarily and the number of connections secondarily.

For this problem, the following values must appear on the < problem-parameter-card >:

Cols. 1 ~ 4 4, the number of external variables

Cols. 5 ~ 8 2, the number of output functions

Cols. 9 ~ 12 8, the number of gates in the original network

Cols. 13 ~ 16 100, the value of A

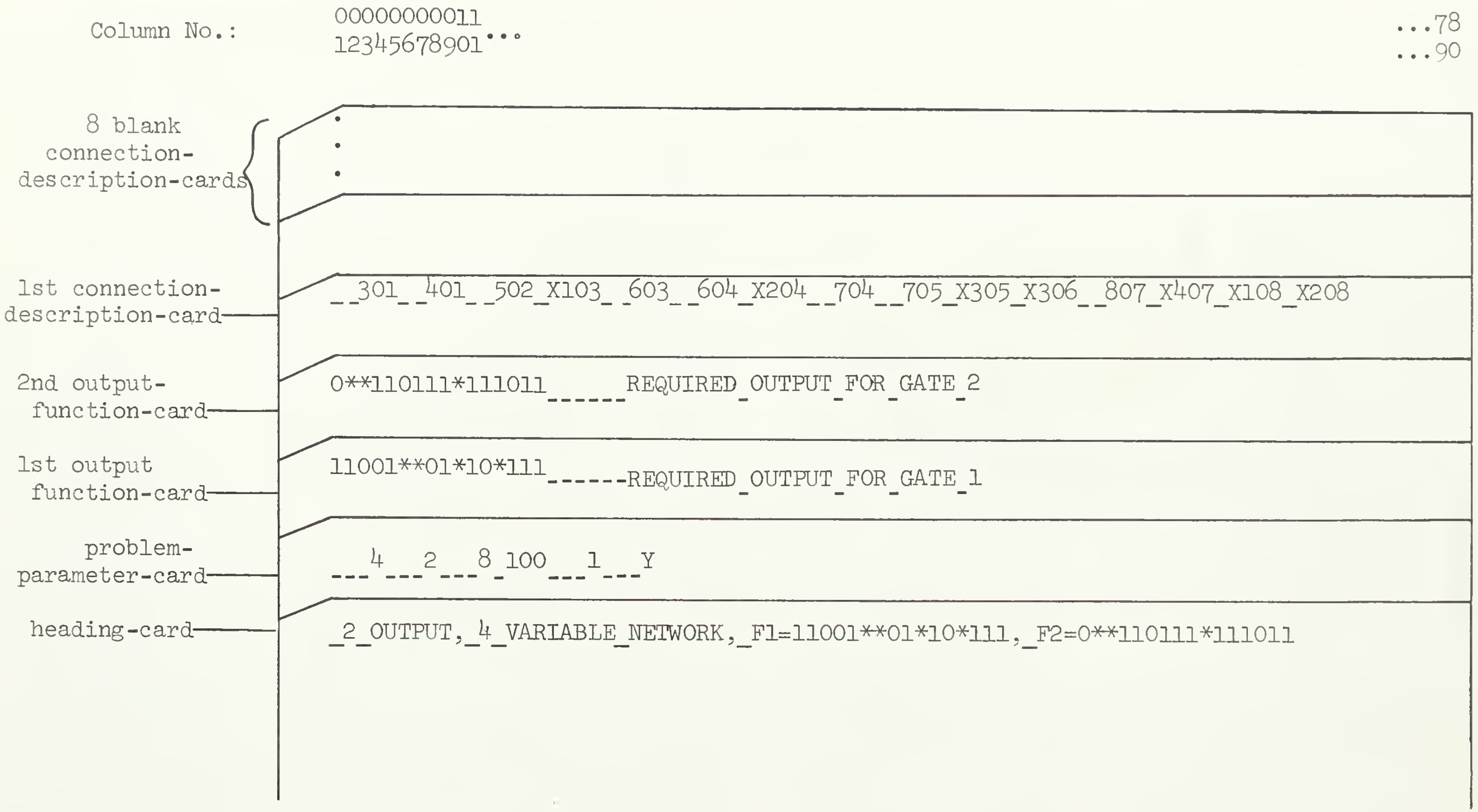
Cols. 17 ~ 20 1, the value of B

Col. 24 Y, indicative of an implicit specification of external variables and the availability of both complemented and uncomplemented variables

Fig. 5.3.3 shows the setup of the data cards corresponding to this problem. Notice the differences and similarities to the data cards shown in Fig. 5.3.2. The < problem-parameter-card > differs only in column 24. The < external-variable-card > s are missing in both Fig. 5.3.2 and Fig. 5.3.3 since the external variables are implicitly specified for both problems. The < output- function-card > s, however, appear in Fig. 5.3.3 but not in 5.3.2 since they are necessary to specify "don't-care" components which do not occur in the completely specified output functions of Example 1. In both cases, though, the < connection-description-card > s are identical since the original networks are identical.

By allowing "don't-care" terms in the output functions, and by allowing the use of both complemented and uncomplemented variables[†] (even though the original network employed only uncomplemented variables), the restrictions during the transduction process are loosened (compared to what they were for Example 1), perhaps permitting a network of

[†] In the case of NETTRA-PG1, -P1, and -P2, it is useless to specify Y rather than X in column 24 for this example. Since the original network uses only uncomplemented variables, to these programs which perform "pruning" procedures (i.e., procedures which are incapable of adding new connections) the availability of complemented variable is not meaningful.



```
__301__401__502_X103__603__604_X204__704__705_X305_X306__807_X407_X108_X208
0**110111*111011______REQUIRED_OUTPUT_FOR_GATE_2
11001**01*10*111------REQUIRED_OUTPUT_FOR_GATE_1
___4___2___8_100___1___Y
_2_OUTPUT,_4_VARIABLE_NETWORK,_F1=11001**01*10*111,_F2=0**110111*111011
```

Fig. 5.3.3 Possible setup of data cards to specify the problem given in Example 2.

less cost to be obtained.

Notice that the first < output-function-card > corresponds to the output of gate 1 and the second < output-function-card > corresponds to the output of gate 2. This must hold true for every problem in which < output-function-card > s are included; the gates labeled 1, 2, ..., m must correspond to the output functions specified on < output-function-card > s 1, 2, ..., m, respectively.

Of course, the printed output of the program will be in the same format described in Example 1.

Example 3: The three output network of six variables shown in Fig. 5.3.4. The outputs are again assumed to be incompletely specified. In fact, only the following 11 input combinations are specified out of a possible 64 ($= 2^6$):

x_1	0	0	0	0	0	0	0	0	0	0	1
x_2	0	0	0	0	0	0	0	1	1	1	0
x_3	0	0	0	0	0	0	0	0	0	1	1
x_4	0	0	0	0	1	1	1	0	0	0	1
x_5	0	0	1	1	0	0	1	0	1	1	0
x_6	0	1	0	1	0	1	0	1	1	0	0
F_1	0	0	1	1	0	0	*	0	0	0	0
F_2	1	1	*	1	1	1	0	1	1	0	*
F_3	1	1	0	0	0	0	0	1	0	0	0

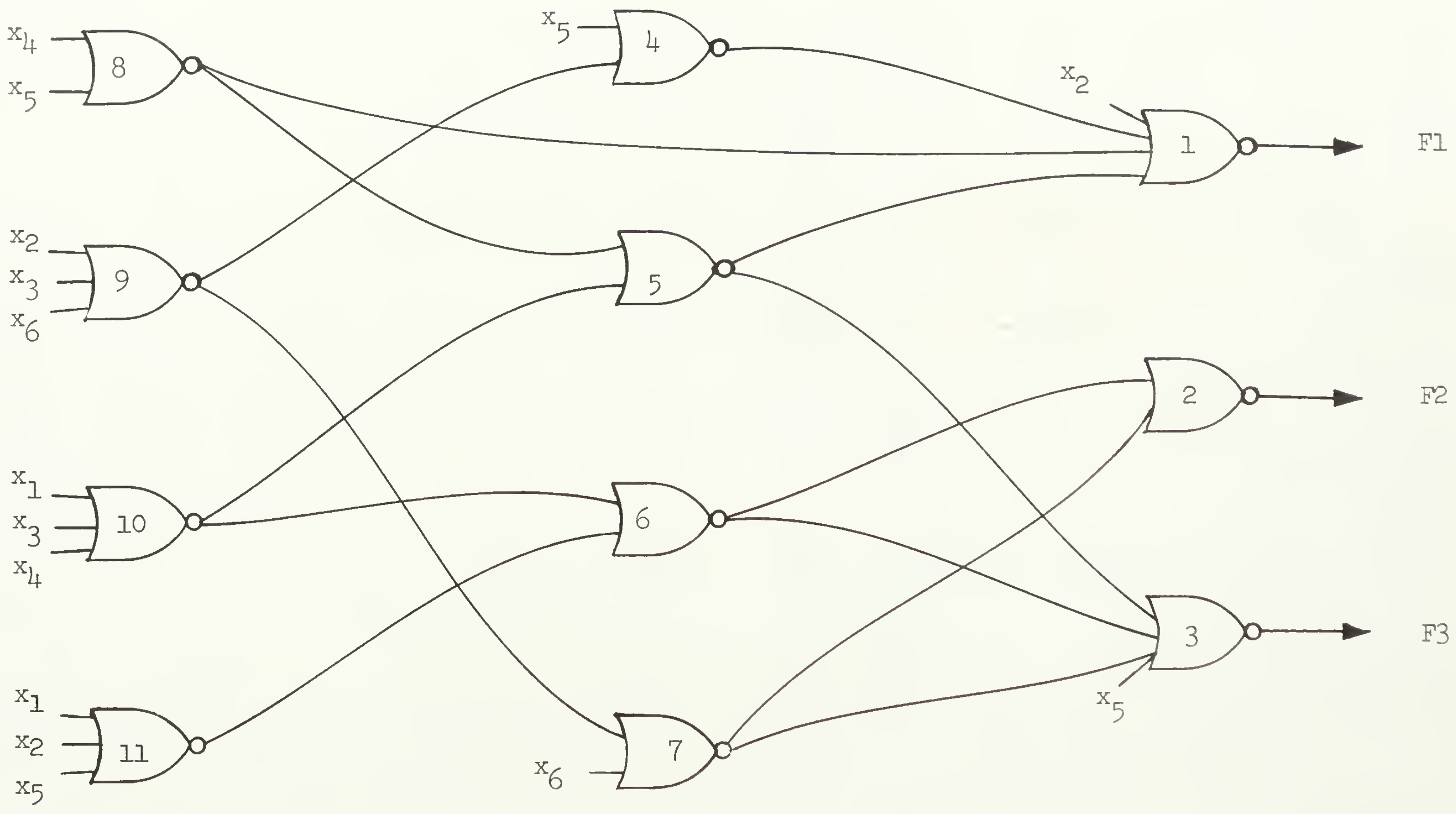


Fig. 5.3.4 Network to be transformed in Example 3.

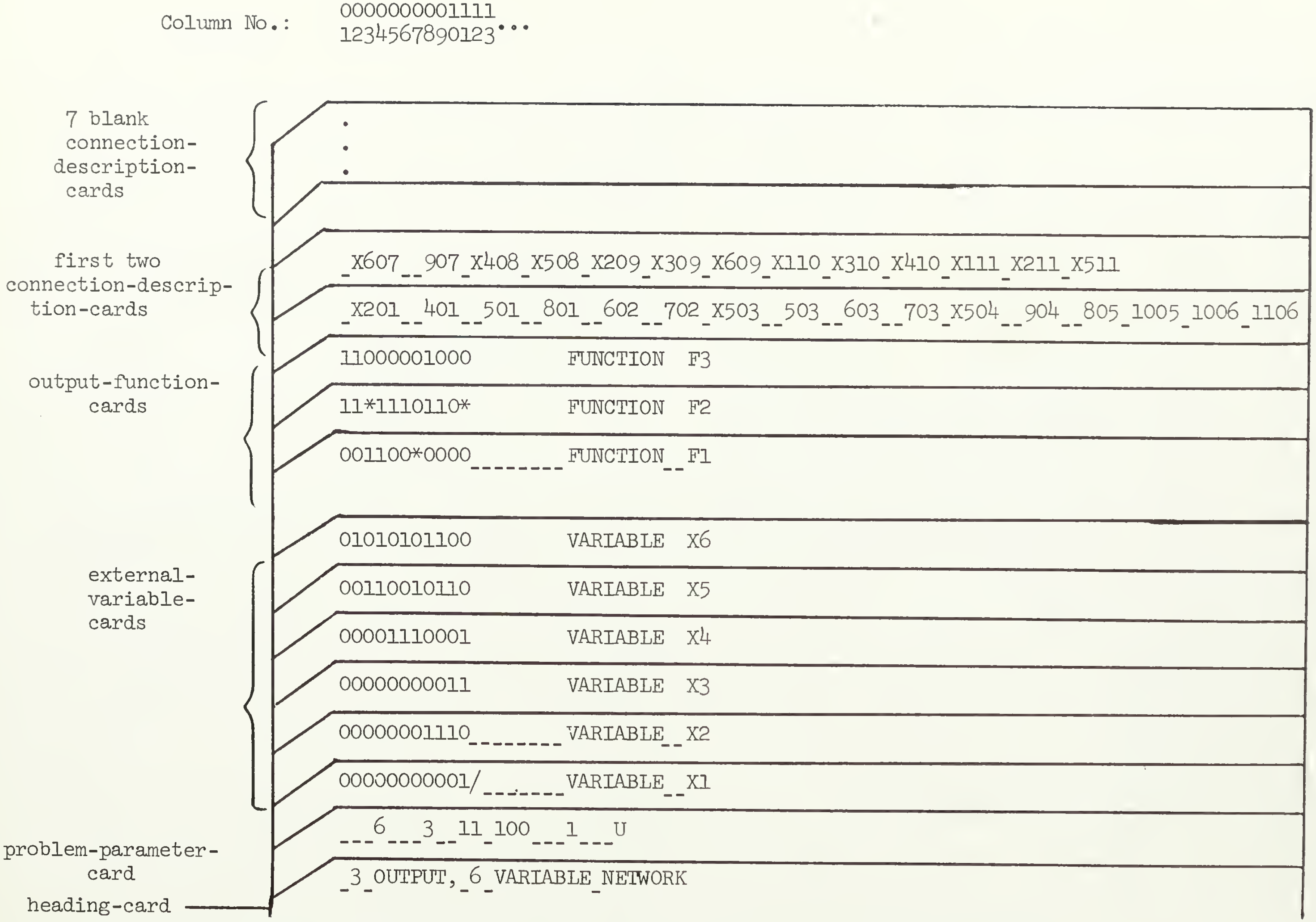
Additionally, only uncomplemented variables are assumed to be available, and the problem is to reduce the number of gates primarily and the number of connections secondarily.

For this example, the following parameters appear on the < problem-parameter-card >:

Cols.	1 ~ 4	6, the number of external variables
Cols.	5 ~ 8	3, the number of output functions
Cols.	9 ~ 12	11, the number of gates in the original network
Cols.	13 ~ 16	100, the value of A
Cols.	17 ~ 20	1, the value of B
Col.	24	U, indicative of an explicit specification of external variables and the availability of only uncomplemented variables

Fig. 5.3.5 shows a possible setup of the data cards corresponding to this example. Notice that in this example, the <external-variable-card > s are included, whereas in the two previous examples they were omitted. Although this problem is not too realistic (none of the 3 functions is actually a 6-variable function), it demonstrates the input data preparation to be used in cases where many external variables are present and a high percentage of "don't care" terms exist.

Again, the printed output from the program will follow the same format described in Example 1.



```
Column No.:  0000000001111
             1234567890123...

.
.
.

_X607__907_X408_X508_X209_X309_X609_X110_X310_X410_X111_X211_X511
_X201__401__501__801__602__702_X503__503__603__703_X504__904__805_1005_1006_1106
11000001000        FUNCTION  F3
11*1110110*        FUNCTION  F2
001100*0000________FUNCTION__F1
01010101100        VARIABLE  X6
00110010110        VARIABLE  X5
00001110001        VARIABLE  X4
00000000011        VARIABLE  X3
00000001110________VARIABLE__X2
00000000001/_______VARIABLE__X1
___6___3__11_100___1___U
_3_OUTPUT,_6_VARIABLE_NETWORK
```

Fig. 5.3.5 Possible setup of data cards to specify the problem given in Example 3.

REFERENCES

[1] J. N. Culliney, "Program manual: NOR network transduction based on connectable and disconnectable conditions (Reference manual of NOR network transduction programs NETTRA-G1 and NETTRA-G2)," Report No. UIUCDCS-R-75-698, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., Feb., 1975.

[2] J. N. Culliney, H. C. Lai, and Y. Kambayashi, "Pruning procedures for NOR networks using permissible functions (Principles of NOR network transduction programs NETTRA-PG1, NETTRA-P1, and NETTRA-P2)," Report No. UIUCDCS-R-74-690, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., Nov. 1974.

[3] Y. Kambayashi and J. N. Culliney, "NOR network transduction procedures based on connectable and disconnectable conditions (Principles of NOR network transduction programs NETTRA-G1 and NETTRA-G2)," to appear as a report, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill.

[4] Y. Kambayashi and S. Muroga, "Network transduction based on permissible functions (General principles of NOR network transduction NETTRA programs)," to appear as a report, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill.

[5] Y. Kambayashi, H. C. Lai, J. N. Culliney, and S. Muroga, "NOR network transduction based on error-compensation (Principles of NOR network transduction programs NETTRA-E1, NETTRA-E2, and NETTRA-E3)," Report No. UIUCDCS-R-75-737, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill.,June 1975.

[6] H. C. Lai, "Program manual: NOR network transduction by generalized gate merging and substitution (Reference manual of NOR network transduction programs NETTRA-G3 and NETTRA-G4)," Report No. UIUCDCS-R-75-714, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., April 1975.

[7] H. C. Lai and J. N. Culliney, "Program manual: NOR network pruning procedures using permissible functions (Reference manual of NOR network transduction programs NETTRA-PG1, NETTRA-P1, and NETTRA-P2)," Report No. UIUCDCS-R-74-686, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., Nov. 1974.

[8] H. C. Lai and Y. Kambayashi, "NOR network transduction by generalized gate merging and substitution (Principles of NOR network transduction programs NETTRA-G3 and NETTRA-G4)," Report No. UIUCDCS-R-75-728, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., June, 1975.

APPENDIX:
Program Listings

Following are the listings of the FORTRAN programs NETTRA-E1, NETTRA-E2, and NETTRA-E3. These programs realize, respectively, the transduction procedures discussed in Section 2, Section 3.1, and Section 3.2.

Since NETTRA-E2 and NETTRA-E3 only differ in a few subroutines from NETTRA-E1, only these subroutines are listed for NETTRA-E2 and -E3: MAIN (differs slightly in each of the three programs), ALPATH (used only in NETTRA-E3, and PROCCE (slightly different for NETTRA-E3).

Explanations of variables used in the programs can be found in the listings themselves.

```
C
C
C*************************************************************************
C                                                                        *
C                                                                        *
C         PPPP    RRRR     OOO     GGG    RRRR      A      M     M       *
C         P   P   R   R   O   O   G   G   R   R    A A     MM   MM       *
C         P   P   R   R   O   O   G       R   R   A   A    M M M M       *
C         PPPP    RRRR    O   O   G  GG   RRRR    AAAAA    M  M  M       *
C         P       R  R    O   O   G   G   R  R    A   A    M     M       *
C         P       R   R    OOO     GGG    R   R   A   A    M     M       *
C                                                                        *
C                                                                        *
C                                                                        *
C N   N   EEEEE   TTTTT   TTTTT   RRRR      A              EEEEE    1    *
C NN  N   E         T       T     R   R    A A             E       11    *
C N N N   E         T       T     R   R   A   A            E        1    *
C N  NN   EEE       T       T     RRRR    AAAAA   XXXXX    EEE      1    *
C N   N   E         T       T     R  R    A   A            E        1    *
C N   N   EEEEE     T       T     R   R   A   A            EEEEE   111   *
C                                                                        *
C                                                                        *
C*************************************************************************
C
C
C     SUBROUTINE MAIN
C     EDITION BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
C
C     NOTE: ALL COMMON VARIBLES MIGHT NOT BE USED IN THIS PROGRAM.
C
C     COMMON VARIABLES:
C         $GT: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY
C              IN THIS COL. TELLS GATE WHERE FN. IS REALIZED.
C        $LTH: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C              IN THIS COL. TELLS HOW MANY CONNECTIONS MUST BE ADDED.
C        $NOE: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C              IN THIS COL. TELLS THE NUMBER OF 1-ERRORS CREATED IF THIS
C              ROW IS USED.
C         $PW: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C              IN THIS COLUMN TELLS THE PREFERENCE WEIGHT.
C           A: WEIGHT FOR NO. OF GATES IN COMPUTING COST FUNCTION.
C           B: WEIGHT FOR NO. OF CONNECTIONS IN COMPUTING COST FUNCTION.
C        COST: COST OF NETWORK - A MEASURE OF NETWORK SIZE.
C       ESS1S: RECORDS NO. OF ESSENTIAL 1'S IN EVERY INPUT TO CURRENT GCO
C              (POSITIONS IN ESS1S CORRES. TO GATES NOT FEEDING GCO ARE
C              IGNORED).
C       F$UB1: POINTS TO LAST ELEMENT IN F$1.
C         F$1: LISTS (CONSECUTIVELY) POSITIONS OF DESIRABLE 1'S (FOR
C              COVERING) IN A CONNECTIBLE FUNCTION.
C          GI: LABEL OF A PARTICULAR GATE.
C      GLEVEL: GLEVEL(GI) TELLS WHICH LEVEL OF THE NETWORK GI IS IN.
C      GSMALL: STORES INTERMEDIATE AND FINAL CALCULATED CSPF'S.
C       HLIST: HLIST(I,J) GIVES NAME OF I-TH GATE (OR EX. VAR.) IN NET-
C              WORK LEVEL J.
C        IDXO: LIST OF O-COORDINATES IN CSPFE OF THE GATE UNDER
C              CONSIDERATION.
C       IDXOE: LIST OF O-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
C              CONSIDERATION.
C        IDX1: LIST OF 1-COORDINATES IN CSPFE OF THE GATE UNDER
C              CONSIDERATION.
C       IDX1E: LIST OF 1-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
```

```
C           CONSIDERATION.                                                  E1 00370
C    IFLAG: SAME AS EYEFLG IN SUBROUTINE PROCII.                            E1 00380
C  INC$MX: INC$MX(GI,GJ)>0 MEANS THERE EXISTS A CONNECTION FROM GATE        E1 00390
C           (OR EX. VAR.) GI TO GATE GJ.  INC$MX(GI,GJ)=0 IF NOT.           E1 00400
C  INPTOV: LISTS FOR EACH CORRESPONDING ENTRY OF F$1, HOW MANY INPUTSE1 00410
C           HAVE A '1' IN THE POSITION INDICATED BY F$1.                    E1 00420
C    IPATH: IPATH(GI)=1 MEANS GATE GI IS ON A PATH FROM A CERTAIN GATEE1 00430
C           TO AN OUTPUT GATE.  OTHERWISE IPATH(GI) = 0.                    E1 00440
C    IPRED: IPRED(I,GJ) GIVES THE NAME OF THE I-TH GATE OR EX. VAR. INE1 00450
C           A LIST OF GATES AND EX. VAR. FEEDING GJ.                        E1 00460
C    ISUCC: ISUCC(I,GJ) GIVES THE NAME OF THE I-TH GATE FED BY GJ.          E1 00470
C    JFLAG: SAME AS JAYFLG IN SUBROUTINE PROCII.                            E1 00480
C     KEYA: A FLAG INDICATING IF ANY ERROR COMPENSATION HAS BEEN            E1 00490
C           PERFORMED.                                                      E1 00500
C     KEYB: A FLAG INDICATING IF ANY PRIMARY 0-ERROR-COORDINATES HAS        E1 00510
C           BEEN COMPENSATED.                                               E1 00520
C    KFLAG: SAME AS KEIFLG IN PROCII.                                       E1 00530
C     LEVM: NUMBER OF LEVELS IN THE NETWORK (NOTE EX. VAR. ARE ALSO         E1 00540
C           ASSIGNED LEVELS JUST LIKE GATES).                               E1 00550
C   LGLIST: LGLIST(J) TELLS NO. OF GATES AND EX. VAR. IN LEVEL J OF         E1 00560
C           NETWORK.                                                        E1 00570
C      LIP: NUMBER OF PREDECESSORS FOR THE GATE UNDER CONSIDERATION.        E1 00580
C   LIPRED: LIPRED(GI) TELLS NO. OF IMMEDIATE PREDECESSORS OF GATE GI.E1 00590
C    LISTC: ORDERED LIST OF CONNECTIBLE INPUTS TO GCO.  ORDERED BY          E1 00600
C           DECREASING NO. OF 0'S IN GCO COVERED.                           E1 00610
C    LISTL: ORDERED LIST OF GATES AND EX. VAR. WHICH ORIGINALLY FED         E1 00620
C           GCO AND WHICH HAVE NOT YET BEEN DISCONNECTED.  ORDERED BY       E1 00630
C           DECREASING NO. OF ESSENTIAL 1'S.                                E1 00640
C   LISUCC: LISUCC(GI) TELLS NO. OF IMMEDIATE SUCCESSORS OF GATE (OR        E1 00650
C           EX. VAR.) GI.                                                   E1 00660
C    LMTS2: UPPER LIMIT OF THE NUMBER OF ELEMENTS IN SET S2.                E1 00670
C   LPOTAB: FOR GATE GI, LPOTAB(GI) POINTS TO LAST ROW OF POTAB             E1 00680
C           CONCERNING GI.                                                  E1 00690
C        M: NUMBER OF NETWORK OUTPUT GATES.                                 E1 00700
C        N: NUMBER OF EXTERNAL VARIABLES (OR INPUT FNC.) AVAILABLE.         E1 00710
C   NEPMAX: FOR ERROR COMPENSATION PROGRAMS.  IF MORE THAN NEPMAX           E1 00720
C           ERROR POSITIONS OCCUR WHEN A PARTICULAR GATE IS REMOVED,        E1 00730
C           PROGRAM SKIPS ATTEMPT TO COMPENSATE FOR THAT GATE'S             E1 00740
C           REMOVAL.  VALUE CAN BE SPECIFIED BY USER, OTHERWISE EQUAL E1 00750
C           TO ONE HALF OF N2 BY DEFAULT.                                   E1 00760
C       NM: SUM OF N PLUS M                                                 E1 00770
C      NM1: SUM OF NM PLUS 1.                                               E1 00780
C      NN2: PRODUCT OF N AND N2.                                            E1 00790
C      NOS: NUMBER OF ELEMENTS IN SET S.                                    E1 00800
C     NOS1: NUMBER OF ELEMENTS IN SET S1.                                   E1 00810
C   NOS1SV: NUMBER OF ELEMENTS IN SET S1 BEFORE ENTERING SUBROUTINE         E1 00820
C           RPLCF.                                                          E1 00830
C     NOS2: NUMBER OF ELEMENTS IN SET S2.                                   E1 00840
C     NOT1: NUMBER OF ELEMENTS IN SET T1.                                   E1 00850
C   NOT1SV: NUMBER OF ELEMENTS IN SET T1 BEFORE ENTERING SUBROUTINE         E1 00860
C           RPLCF.                                                          E1 00870
C      NO0: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0.                        E1 00880
C     NO0E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0E.                       E1 00890
C      NO1: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1.                        E1 00900
C     NO1E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1E.                       E1 00910
C       NR: SUM OF N PLUS R.                                                E1 00920
C     NRN2: PRODUCT OF NR AND N2.                                           E1 00930
C    NRPLC: NRPLC(I) STORES THE NUMBER OF ELEMENTS IN RPLC(I,*)             E1 00940
C                                                         FOR I=1,2.        E1 00950
C       N1: SUM OF N PLUS 1.                                                E1 00960
C       N2: NUMBER OF DIFFERENT INPUT COMBINATIONS TO BE CONSIDERED         E1 00970
```

```
C           (USUALLY 2 TO THE POWER N).                                  E1 00980
C     ORIGIN: ORIGIN(GI)=1 MEANS GI ORIGINALLY CONNECTED TO GCO.         E1 00990
C             ORIGIN(GI)=0 MEANS GI DID NOT FEED GCO ORIGINALLY.         E1 01000
C         P$: P$(1,-) CONSECUTIVELY LISTS OUTPUTS OF EVERY EX. VAR. AND E1 01010
C             EVERY GATE (FOR EVERY INPUT COMBINATION): P$(1,1),....,    E1 01020
C             P$(1,N2) FOR FIRST EX VAR; P$(1,N2+1),....,P$(1,2*N2) FOR  E1 01030
C             SECOND EX VAR; ... ; P$(1,N*N2+1),...., P$(1,N*N2+N2) FOR  E1 01040
C             FIRST GATE; ETC.  P$(2,-) IS USED AS WORK SPACE FOR        E1 01050
C             CALCULATIONS ASSOCIATED WITH P$(1,-).                      E1 01060
C        PCO: FOR ERROR COMPENSATION PROCEDURES.  PCO IS THE GATE        E1 01070
C             REMOVED FROM ORIGINAL NETWORK TO OBTAIN CURRENT ALTERED    E1 01080
C             NETWORK.                                                   E1 01090
C     POINTA: NOT USED.                                                  E1 01100
C     POINTC: POINTS TO LAST ELEMENT IN LISTC.                           E1 01110
C     POINTL: POINTS TO LAST ELEMENT IN LISTL.                           E1 01120
C     POINTR: POINTS TO LAST ELEMENT IN RNEC1 (IN SUBROUTINE SUBSTI).    E1 01130
C      POTAB: POTENTIAL OUTPUT TABLE.  HOLDS INFORMATION ABOUT ALL       E1 01140
C             COMBINATIONS OF CONNECTIONS TO FORM NEW (AND HOPEFULLY     E1 01150
C             USEFUL) FUNCTIONS.                                         E1 01160
C     PPOTAB: FOR GATE GI, PPOTAB(GI) POINTS TO FIRST OF A SEQUENCE OF   E1 01170
C             ROWS OF POTAB CONCERNING GI.                               E1 01180
C          R: NUMBER OF GATES IN THE NETWORK (EXCLUDES EX VAR, ALSO      E1 01190
C             NOTE SOME OF R GATES MAY BE ISOLATED).                     E1 01200
C       RPLC: RPLC(1,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE   E1 01210
C                       ERROR-COORDINATES OF WEIGHT 2 OR ABOVE.          E1 01220
C             RPLC(2,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE   E1 01230
C                       AT LEAST ONE ERROR-COORDINATE OF WEIGHT 1.       E1 01240
C     RSCONN: LIST OF CONNECTIONS ADDED TO A NETWORK (IN CODED FORM).    E1 01250
C     RTCONN: LIST OF CONNECTIONS REMOVED FROM A NETWORK (CODED FORM).   E1 01260
C          S: NO. OF CONNECTIONS ADDED TO A NETWORK.  POINTS TO LAST     E1 01270
C             ENTRY IN RSCONN.                                           E1 01280
C       SETS: SET S CONSISTING OF INPUTS OF THE GATE UNDER CONSIDERATIONE1 01290
C             WHICH ARE TO BE REPLACED IF POSSIBLE.                      E1 01300
C      SETS1: SET S1 CONSISTING OF ELEMENTS OF SET S WHICH CAN BE        E1 01310
C             REPLACED BY ELEMENTS IN SET S2.                            E1 01320
C      SETS2: SET S2 CONSISTING OF FUNCTIONS WHICH ARE CANDIDATES FOR    E1 01330
C             REPLACING ELEMENTS IN SET S.                               E1 01340
C      SETT1: SET T1 CONSISTING OF ESSENTIAL ONES COVERED BY ELEMENTS INE1 01350
C                                                               SET S1.  E1 01360
C        STS: STARTING ELEMENT OF SET S.                                 E1 01370
C     SUC$MX: SUC$MX(GI,GJ)>0 MEANS GATE GJ IS A SUCCESSOR OF GATE GI.   E1 01380
C             SUC$MX(GI,GJ)=0 IF NOT.                                    E1 01390
C       SUMP: SUM OF ALL ACTIVE INPUTS OF THE GATE UNDER CONSIDERATION.  E1 01400
C      SUMS2: SUM OF ALL ACTIVE ELEMENTS OF SET S2.                      E1 01410
C          T: NUMBER OF CONNECTIONS REMOVED FROM A NETWORK.  POINTS TO   E1 01420
C             LAST ENTRY IN RTCONN.                                      E1 01430
C       TIME: USED TO STORE AMOUNT OF ELAPSED COMPUTATION TIME.          E1 01440
C      UNAME: MNEMONIC NAMES FOR EXTERNAL VARIABLES AND GATES.           E1 01450
C     VF$UB1: POINTS TO LAST ELEMENT IN VF$1.                            E1 01460
C       VF$1: SIMILAR TO F$1, EXCEPT THIS LISTS JUST COMPONENT POSITIONSE1 01470
C             (OF O'S IN CSPF VECTOR OF GCO) COVERED ONLY BY REMAINING   E1 01480
C             ORIGINALLY CONNECTED INPUTS TO GCO.                        E1 01490
C                                                                        E1 01500
C                                                                        E1 01510
C                                                                        E1 01520
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                             E1 01530
      COMMON NERMAX                                                      E1 01540
      COMMON    N            , M            , A            , B           E1 01550
     1     ,    R            , N2           , N1           , NR          E1 01560
     2     ,    NM           , KFLAG        , JFLAG        , COST        E1 01570
     3     ,    LEVM         , NRN2         , NM1          , NN2         E1 01580
```

```
      COMMON    ISUCC(40,40) , LISUCC(40)    , IPRED(40,40) , LIPRED(40)  E1 01590
     1     ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)   E1 01600
     2     ,    GLEVEL(40)   , LGLIST(40)    , HLIST(40,40) , TIME        E1 01610
      COMMON    T            , RTCONN(100)   , S            , RSCONN(100) E1 01620
      COMMON    IFLAG        ,POINTA         ,ESSIS(40)     ,F$1(32)      E1 01630
     1    ,F$UB1             ,INPTCV(32)     ,LISTC(40)     ,POINTC       E1 01640
     2    ,LISTL(40)         ,POINTL         ,ORIGIN(40)    ,IPATH(40)    E1 01650
     3    ,POINTR            ,VF$1(32)       ,VF$UB1        ,GSMALL(40,32)E1 01660
      COMMON    POTAB(200,42),PPOTAB(40)     ,LPOTAB(40)    ,NRPLC(2)     E1 01670
     1    ,RPLC(2,40)        ,IDX0(32)       ,IDX0E(32)     ,IDX1(32)     E1 01680
     2    ,IDX1E(32)         ,SUMP(32)       ,SETT1(32)     ,NOT1         E1 01690
     3    ,SETS1(40)         ,NOS1           ,SETS(40)      ,NOS          E1 01700
     4    ,STS               ,SUMS2(32)      ,SETS2(200)    ,NOS2         E1 01710
     5    ,LIP               ,NOOF           ,KEYA          ,KEYB         E1 01720
     6    ,NOO               ,NO1            ,NO1E          ,$GT          E1 01730
     7    ,$LTH              ,$PW            ,$NOE          ,GI           E1 01740
      COMMON                 NOTISV        ,NOS1SV         ,LMTS2         E1 01750
      DIMENSION CNTLIS(144),UGATE(40),UHEAD(20)                           E1 01760
      DATA KOUNT5 /0/, UBLANK/'    '/                                     E1 01770
  990 READ(5,1000,END=500) UHEAD, N, M, R, A, B, UC, NEPMAX               E1 01780
C     NEPMAX IS THE MAXIMUM ALLOWABLE NUMBER OF ERROR POSITIONS           E1 01790
 1000 FORMAT(20A4/5I4,A4,I4)                                              E1 01800
      KEYXC=0                                                             E1 01810
      IF(UC.NE.UBLANK) KEYXC=1                                            E1 01820
      CALL PAGE                                                           E1 01830
      CALL LINE(10)                                                       E1 01840
      KOUNT5=KOUNT5+1                                                     E1 01850
      PRINT 2, KOUNT5                                                     E1 01860
    2 FORMAT(20X,'*** OPTIMAL NOR NETWORK ***',50X,'PROBLEM NO.= ',I4 ) E1 01870
      CALL LINE(4)                                                        E1 01880
      PRINT 1005, UHEAD                                                   E1 01890
 1005 FORMAT(25X,20A4)                                                    E1 01900
      CALL LINE(4)                                                        E1 01910
      PRINT 10, N,M,A,B                                                   E1 01920
   10 FORMAT(30X,'NUMBER OF VARIABLES =',I4 //                            E1 01930
     1        30X,'NUMBER OF FUNCTIONS =',I4 //                           E1 01940
     2        30X,'COST COEFFICIENT A  =',I4//                            E1 01950
     3        47X,                 'B  =',I4)                             E1 01960
      CALL LINE(1)                                                        E1 01970
      IF(KEYXC.NE.0) GO TO 25                                             E1 01980
      PRINT 21                                                            E1 01990
   21 FORMAT(1H0,29X,'--- UNCOMPLEMENTED VARIABLES  X ---')               E1 02000
      GO TO 30                                                            E1 02010
   25 CONTINUE                                                            E1 02020
      PRINT 28                                                            E1 02030
   28 FORMAT(1H0,29X,'--- BOTH COMPLEMENTED AND UNCOMPLEMENTED VARIABLESE1 02040
     1 X, Y ---')                                                         E1 02050
   30 CONTINUE                                                            E1 02060
      CALL LINE(5)                                                        E1 02070
C***** SET UP EXTERNAL VARIABLES *****                                    E1 02080
      N2=2**N                                                             E1 02090
      IF(NEPMAX.EQ.0)NEPMAX = N2/2                                        E1 02100
      H=N*N2                                                              E1 02110
      J=N2                                                                E1 02120
      L= 1                                                                E1 02130
      I=0                                                                 E1 02140
      DO 1011 II=1,N                                                      E1 02150
       J=J/2                                                              E1 02160
       L=L*2                                                              E1 02170
       SN= 1                                                              E1 02180
       DO 1010 LL=1,L                                                     E1 02190
```

```
        SN=-SN
        V=(1+SN)/2
        DO 1009 JJ=1,J
         I=I+1
         P$(1,I)=V
      IF(KEYXC.NE.0)P$(1,I+H)=1-V
 1009   CONTINUE
 1010  CONTINUE
 1011 CONTINUE
      IF(KEYXC.NE.0) N=N+N
      N1=N+1
      NM=N+M
      NM1=NM+1
      NN2=N*N2+1
      NR=N+R
      NRN2=NR*N2
      CALL OUTPUT(INC$MX,KEYXC)
C***** READ IN NETWORK INFORMATION AND SET UP INC$MX *****
      READ 1001,   CNTLIS
 1001 FORMAT(16I5)
      DO 1115 GI=1,NR
      DO 1115 GJ=1,NR
 1115 INC$MX(GI,GJ)=0
      DO 1120 I=1,144
       ITEM=CNTLIS(I)
      IF(ITEM.EQ.0) GO TO 1119
       GI=ITEM/100
       GJ=ITEM-100*GI
       INC$MX(GI,GJ)=1
       GO TO 1120
 1119 COST=A*R+B*(I-1)
       GO TO 1130
 1120 CONTINUE
 1130 CONTINUE
      CALL SUBNET
      CALL PVALUE
      CALL LINE(4)
      PRINT 1140, COST
 1140 FORMAT(20X,' ORIGINAL NETWORK    COST=', I5)
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
C
C***** ENTRY REDUNDANCY CHECK *****
      S = 0
      T = 0
      CALL UNNECE
      GATES = M
      C = 0
      DO 4 GI =  1,NR
      C = C + LISUCC(GI)
      IF(GI.LE.NM)GOTO4
      IF(LISUCC(GI).GT.0)GATES=GATES+1
    4 CONTINUE
      OLDCST = A*GATES + B*(C)
      T=0
      S=0
C     INITIALIZE TIMER TO 10 MINUTES
      CALL STIMEZ(60000)
      TIME = KTIMEZ(0)
```

```
C****   PROCEDURE  PROCCE                                                E1 02810
      CALL PROCCE(WORKED)                                                E1 02820
C     CALL FOR ELAPSED TIME                                              E1 02830
      TIME = KTIMEZ(0) - TIME                                            E1 02840
      CALL LINE(4)                                                       E1 02850
      PRINT 3915                                                         E1 02860
 3916 FORMAT(20X,'TIME ELAPSED =',I8,'  CENTISECONDS')                   E1 02870
 3915 FORMAT(20X,'NETWORK DERIVED BY PROCCE')                            E1 02880
      PRINT 3916,TIME                                                    E1 02890
      CALL LINE(4)                                                       E1 02900
      CALL TRUTH(P$,1)                                                   E1 02910
      CALL LINE(4)                                                       E1 02920
      CALL CKT(INC$MX,GLEVEL)                                            E1 02930
      GATES = M                                                          E1 02940
      C = 0                                                              E1 02950
      DO 36 GI = 1,NR                                                    E1 02960
      C = C + LISUCC(GI)                                                 E1 02970
      IF(GI.LE.NM) GO TO 36                                              E1 02980
      IF(LISUCC(GI).GT.0) GATES = GATES + 1                              E1 02990
   36 CONTINUE                                                           E1 03000
      NEWCST = A*GATES + B*C                                             E1 03010
      IF(NEWCST.LT.OLDCST)GO TO 37                                       E1 03020
      PRINT 105                                                          E1 03030
  105 FORMAT(1H ,10X,'NO REDUNDANCY FOUND.')                             E1 03040
      GO TO 990                                                          E1 03050
   37 CALL LINE(3)                                                       E1 03060
      PRINT 320,NEWCST                                                   E1 03070
  320 FORMAT(9X,'* A NETWORK DERIVED BY PROCCE'/9X,' COST=',I5,'.')      E1 03080
      GO TO 990                                                          E1 03090
  500 STOP                                                               E1 03100
       END                                                               E1 03110


      SUBROUTINE CALS1                                                   E1 03120
C                                                                        E1 03130
C**** THIS SUBROUTINE CALCULATES A MAXIMUM SUBSET, S1, OF S               E1 03140
C      WHICH CONTAINS FUNCTIONS REPLACEABLE  BY S2.  THE NUMBER          E1 03150
C      OF INITIAL ELEMENTS IN S1 (NOS1) IS SET BY CALLING PROGRAM.       E1 03160
C      THE NUMBER OF ESSENTIAL ONES COVERED BY INITIAL S1 IS ALSO        E1 03170
C      SET BY CALLING PROGRAM (NCT1 AND SETT1(*))                        E1 03180
C                                                                        E1 03190
C                                                                        E1 03200
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.    E1 03210
C                                                                        E1 03220
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                             E1 03230
      COMMON NEPMAX                                                      E1 03240
      COMMON   N              , M              , A              , B      E1 03250
     1      ,  R              , N2             , N1             , NR     E1 03260
     2      ,  NM             , KFLAG          , JFLAG          , COST   E1 03270
     3      ,  LEVM           , NRN2           , NM1            , NN2    E1 03280
      COMMON   ISUCC(40,40)   , LISUCC(40)     , IPRED(40,40)   , LIPRED(40) E1 03290
     1      ,  INC$MX(40,40)  , SUC$MX(40,40)  , P$(2,1280)     , UNAME(40)  E1 03300
     2      ,  GLEVEL(40)     , LGLIST(40)     , HLIST(40,40)   , TIME   E1 03310
      COMMON   T              , RTCONN(100)    , S              , RSCONN(100) E1 03320
      COMMON   IFLAG          ,POINTA          ,ESS1S(40)       ,F$1(32) E1 03330
     1      ,F$UB1            ,INPTCV(32)      ,LISTC(40)       ,POINTC  E1 03340
     2      ,LISTL(40)        ,POINTL          ,ORIGIN(40)      ,IPATH(40) E1 03350
     3      ,POINTR           ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)E1 03360
      COMMON   POTAB(200,42),PPOTAB(40)        ,LPOTAB(40)      ,NRPLC(2)  E1 03370
     1      ,RPLC(2,40)       ,IDX0(32)        ,IDX0E(32)       ,IDX1(32)  E1 03380
     2      ,IDX1E(32)        ,SUMP(32)        ,SETT1(32)       ,NOT1      E1 03390
     3      ,SETS1(40)          ,NOS1               ,SETS(40)          ,NOS
     4      ,STS                ,SUMS2(32)          ,SETS2(200)        ,NOS2
     5      ,LIP                ,NO0E               ,KEYA              ,KEYB
     6      ,NO0                ,NO1                ,NO1E              ,$GT
     7      ,$LTH               ,$PW                ,$NOE              ,GI
      COMMON                    NOT1SV             ,NOS1SV            ,LMTS2
      NOT1=NOT1SV
      NOS1=NOS1SV
C     STS IS THE STARTING ELEMENT OF SETS
      DO 7800 NO = STS,NOS
        GP = SETS(NO)
C****   GP > 1000 : ALREADY REMOVED *****
        IF(GP.GT.1000)GO TO 7800
        BSGP = (GP-1)*N2
C****   CALCULATE ESSENTIAL ONES IN GP AND CHECK WHETHER SETS2 COVERS
C****   THEM OR NOT *****
        NOTO = NOT1
        DO 7100 NOZ=1,NO0
          TH = IDXO(NOZ)
          IF(SUMP(TH).NE.1.OR.P$(1,BSGP+TH).NE.1)GO TO 7100
          IF(SUMS2(TH).LE.0)GO TO 7800
          NOTO = NOTO + 1
          SETT1(NOTO) = TH
 7100   CONTINUE
        NOT1 = NOTO
        NOS1 = NOS1 + 1
        SETS1(NOS1) = GP
C****   UPDATE SUMP *****
        DO 7300 TH=1,N2
          SUMP(TH) = SUMP(TH) - P$(1,BSGP+TH)
 7300   CONTINUE
C****   UPDATE SET S (MAKE GP INACTIVE) *****
        SETS(NO ) =1000 + GP
 7800 CONTINUE
      RETURN
      END


      SUBROUTINE CONECT(PTR)
C
C**** THIS SUBROUTINE CONNECTS THE FUNCTION IN POTAB SPECIFIED BY PTR
C      TO GATE GI AND MAKES ALL OTHER NECESSARY CONNECTIONS FOR
C      REALIZING THIS FUNCTION.
C
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
      COMMON NEPMAX
      COMMON    N               , M                , A               , B
     1     ,    R               , N2               , N1              , NR
     2     ,    NM              , KFLAG            , JFLAG           , COST
     3     ,    LEVM            , NRN2             , NM1             , NN2
      COMMON    ISUCC(40,40) , LISUCC(40)     , IPRED(40,40) , LIPRED(40)
     1     ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)
     2     ,    GLEVEL(40)    , LGLIST(40)     , HLIST(40,40) , TIME
      COMMON    T               , RTCONN(100)      , S               , RSCONN(100)
      COMMON    IFLAG              ,POINTA             ,ESS1S(40)         ,F$1(32)
     1      ,F$UB1              ,INPTCV(32)         ,LISTC(40)         ,POINTC
     2      ,LISTL(40)          ,POINTL             ,ORIGIN(40)        ,IPATH(40)
     3      ,POINTR             ,VF$1(32)           ,VF$UB1            ,GSMALL(40,32)
```

```
      COMMON    POTAB(200,42),PPOTAB(40)     ,LPOTAB(40)     ,NRPLC(2)
     1     ,RPLC(2,40)      ,IDXO(32)        ,IDXOE(32)      ,IDX1(32)
     2     ,IDX1E(32)       ,SUMP(32)        ,SETT1(32)      ,NOT1
     3     ,SETS1(40)       ,NOS1            ,SETS(40)       ,NOS
     4     ,STS             ,SUMS2(32)       ,SETS2(200)     ,NOS2
     5     ,LIP             ,NODE            ,KEYA           ,KEYB
     6     ,NOO             ,NO1             ,NO1E           ,$GT
     7     ,$LTH            ,$PW             ,$NOE           ,GI
      COMMON                NOT1SV          ,NOS1SV         ,LMTS2
C**** CONNECT THIS FUNCTION
      GP = POTAB(PTR,$GT)
      INC$MX(GP,GI) = 1
      S = S + 1
      RSCONN(S) = 100*GP + GI
C**** CONNECT OTHER NECESSARY CONNECTIONS
      IF(POTAB(PTR,$LTH).EQ.0)GO TO 7200
      LTH = POTAB(PTR,$LTH)
      DO 7100 TH=1,LTH
        GQ = POTAB(PTR,$LTH+TH)
        INC$MX(GQ,GP) = 1
        S = S + 1
        RSCONN(S) = 100*GQ + GP
 7100 CONTINUE
 7200 KEYA = 1
      RETURN
      END


      SUBROUTINE FORC(GJ)
C
C**** REMOVE FIRST ORDER REDUNDANT CONNECTION ******
C            -      -      -         -
C     SUBROUTINE TO REMOVE FIRST ORDER REDUNDANT CONNECTION
C     GJ TO GI IS THE CONNECTION TO BE CHECKED.  INPUT SUM OF GATE GI
C     IS STORED AT SUMP.
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
      COMMON NEPMAX
      COMMON    N             , M              , A              , B
     1    ,     R             , N2             , N1             , NR
     2    ,     NM            , KFLAG          , JFLAG          , COST
     3    ,     LEVM          , NRN2           , NM1            , NN2
      COMMON    ISUCC(40,40)  , LISUCC(40)     , IPRED(40,40)   , LIPRED(40)
     1    ,     INC$MX(40,40) , SUC$MX(40,40)  , P$(2,1280)     , UNAME(40)
     2    ,     GLEVEL(40)    , LGLIST(40)     , HLIST(40,40)   , TIME
      COMMON    T             , RTCONN(100)    , S              , RSCONN(100)
      COMMON    IFLAG           ,POINTA          ,ESS1S(40)      ,F$1(32)
     1     ,F$UB1           ,INPTSV(32)      ,LISTC(40)      ,POINTC
     2     ,LISTL(40)       ,POINTL          ,ORIGIN(40)     ,IPATH(40)
     3     ,POINTR          ,VF$1(32)        ,VF$UB1         ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)     ,LPOTAB(40)     ,NRPLC(2)
     1     ,RPLC(2,40)      ,IDXO(32)        ,IDXOE(32)      ,IDX1(32)
     2     ,IDX1E(32)       ,SUMP(32)        ,SETT1(32)      ,NOT1
     3     ,SETS1(40)       ,NOS1            ,SETS(40)       ,NOS
     4     ,STS             ,SUMS2(32)       ,SETS2(200)     ,NOS2
     5     ,LIP             ,NODE            ,KEYA           ,KEYB
     6     ,NOO             ,NO1             ,NO1E           ,$GT
     7     ,$LTH            ,$PW             ,$NOE           ,GI
      COMMON                NOT1SV          ,NOS1SV         ,LMTS2
```

```
C
C**** LIST ESSENTIAL ONES OF GJ TO GI
      KEYAA = 0
      BSGJ = (GJ-1)*N2
      DO 310 TH=1,N2
        IF(SUMP(TH).NE.1.OR.P$(1,BSGJ+TH).NE.1)GO TO 310
        IF(GSMALL(GI,TH).GT.-1000)GO TO 300
        KEYAA = 1
        GO TO 310
  300   IF(GSMALL(GI,TH).LT.0) RETURN
  310   CONTINUE
C**** DISCONNECT GJ TO GI, UPDATE SUMP *****
      INC$MX(GJ,GI) = 0
      T = T + 1
      RTCONN(T) = 100*GJ + GI
      DO 320 TH=1,N2
  320   SUMP(TH) = SUMP(TH) - P$(1,BSGJ+TH)
      IF(KEYAA.EQ.1) KEYA = 1
      KEYB = 1
      RETURN
      END


      SUBROUTINE MINI2(IMPROV)
C     EDITION AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
C     THE NAME ATTEMPTS TO INDICATE THAT THIS SUBROUTINE IS A MINIATURE
C     VERSION OF PROCEDURE II (PROCII) - ACTUALLY, THIS ROUTINE ONLY
C     REMOVES CONNECTIONS, NONE ARE ADDED
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
C     VARIABLE DEFINITIONS:
C     BESTSL: NAME OF A PRIORITY CANDIDATE TO DISCONNECT FROM GATE GCO.
C     CHOICE: NAME OF A GATE CHOSEN TO BECOME A COVER.
C     COMPNT: A COMPONENT OF AN INTERMEDIATE CSPF VECTOR.
C      EFLAG: SIGNALS WHICH ENTRY POINT USED.
C     FEEDGT: A GATE FEEDING GATE 'GATE'.
C      F$UBO: NUMBER OF 'NECESSARY ZEROS' LISTED IN F$0.
C        F$0: LISTS (CONSECUTIVELY) POSITIONS OF NECESSARY ZEROS IN A
C             CONNECTABLE FUNCTION VECTOR.
C       GATE: NAME OF A GATE.
C     GCOUNT: A COUNTER.
C     GORDER: A SPECIAL ORDERING OF GATES AND EXTERNAL VARIABLES SUCH
C             THAT NO GATE SUCCEEDS A PREDECESSOR IN THE ORDERING.
C     MARKED: MARKED(GI)=1 FOR GI FEEDING 'GATE' INDICATES THAT GI HAS
C             ALREADY BEEN ASSIGNED NECESSARY ZEROS CORRESPONDING TO
C             '1' COMPONENTS IN THE CSPF VECTOR FOR 'GATE'.
C     NMINLV: NUMBER OF GATES IN A CERTAIN LEVEL OF THE NETWORK.
C     SELECT: NAME OF AN INPUT SELECTED AS A CANDIDATE FOR DISCONNECTION
C             FROM GATE 'GCO'.
C          T: COUNTS REMOVED CONNECTIONS.
C     TORDER: A SPECIAL ORDERING OF GATES AND EXTERNAL VARIABLES SUCH
C             THAT VARIABLES COME FIRST FOLLOWED BY GATES WITH DECREASED
C             NUMBERS OF OUTPUTS (TIES ARE BROKEN BY GORDER).
C     TPOINT: POINTER TO TORDER.
C     T1PRED: LIST OF GCO'S PREDECESSORS AT ONE STAGE OF COMPUTATION.
C     T2PRED: LIST OF GCO'S PREDECESSORS AT ONE STAGE OF COMPUTATION.
C      T1SUB: A POINTER TO T1PRED.
C      T2SUB: A POINTER TO T2PRED.
C       USED: USED(GI)=1 MEANS GI IS AN OUTPUT GATE, OR IS A COVER FOR
C             SOME 0-COMPONENT OF 'GATE'. (IT ALSO HAS A TEMPORARY USE
C             IN BEGINNING OF PROGRAM.)
C
C     COUNT,I,II,J,K,L,MOST,Q,TCOUNT,X,XX,Y  ARE USED AS JUST TEMPORARY
C                                                              VARIABLES.
C     HOW TO INCREASE CAPACITY OF SUBROUTINE.
C     DIMENSION: T1PRED(X),T2PRED(X),GORDER(X),
C                MARKED(X),USED(X),TORDER(X)     - X EQUAL TO MAX NUMBER
C                                                  OF GATES PLUS EXTERNAL
C                                                  VARIABLES.
C                F$0(Y) - Y EQUAL TO: 2**(MAX ALLOWED NO. OF EX. VAR.)
C
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
      COMMON NEPMAX
      COMMON    N              , M             , A             , B
     1      ,   R              , N2            , N1            , NR
     2      ,   NM             , KFLAG         , JFLAG         , COST
     3      ,   LEVM           , NRN2          , NM1           , NN2
      COMMON    ISUCC(40,40)   , LISUCC(40)    , IPRED(40,40)  , LIPRED(40)
     1      ,   INC$MX(40,40), SUC$MX(40,40), P$(2,1280)     , UNAME(40)
     2      ,   GLEVEL(40)     , LGLIST(40)    , HLIST(40,40)  , TIME
      COMMON    T              , RTCONN(100)   , S             , RSCONN(100)
      COMMON    IFLAG          ,POINTA         ,ESS1S(40)      ,F$1(32)
     1      ,F$UB1             ,INPTCV(32)     ,LISTC(40)      ,POINTC
     2      ,LISTL(40)         ,POINTL         ,ORIGIN(40)     ,IPATH(40)
     3      ,POINTR            ,VF$1(32)       ,VF$UB1         ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)       ,LPOTAB(40)     ,NRPLC(2)
     1      ,RPLC(2,40)        ,IDXO(32)       ,IDXOE(32)      ,IDX1(32)
     2      ,IDX1E(32)         ,SUMP(32)       ,SETT1(32)      ,NOT1
     3      ,SETS1(40)         ,NOS1           ,SETS(40)       ,NOS
     4      ,STS               ,SUMS2(32)      ,SETS2(200)     ,NOS2
     5      ,LIP               ,NOOE           ,KEYA           ,KEYB
     6      ,NCO               ,NO1            ,NO1E           ,$GT
     7      ,$LTH              ,$PW            ,$NOE           ,GI
      COMMON                   NOT1SV         ,NOS1SV        ,LMTS2
      DIMENSION T1PRED(40),T2PRED(40),GORDER(40),F$0(32),MARKED(40)
      DIMENSION USED(40),TORDER(40)
      IMPROV = 0
      T = 0
C     ORDER GATES IN GORDER
      EFLAG = 0
      GO TO 63
C     THIS ENTRY POINT FOR CALCULATION OF GORDER ONLY
      ENTRY FORMGO
      EFLAG = 1
   63 CONTINUE
      COUNT = 0
      DO 1 I=1,LEVM
      NMINLV = LGLIST(I)
      IF(NMINLV.EQ.0)GOTO1
      DO 2 J=1,NMINLV
      COUNT = COUNT + 1
      GORDER(COUNT) = HLIST(J,I)
    2 CONTINUE
    1 CONTINUE
      IF(EFLAG.EQ.1)RETURN
C     CALCULATE NUMBER OF OUTPUTS OF EACH GATE
C     (THE ARRAY 'USED' IS USED HERE JUST TEMPORARILY)
      DO 51 I=N1,NR
      TCOUNT = 0
      DO 52 J=1,NR
      IF(INC$MX(I,J).EQ.1)TCOUNT = TCOUNT + 1
   52 CONTINUE
C     TCOUNT NOW CONTAINS  THE NUMBER OF OUTPUTS OF GATE I
      USED(I) = TCOUNT
   51 CONTINUE
      MOST = 0
      DO 53 I =N1,NR
      IF(USED(I).GT.MOST)MOST = USED(I)
   53 CONTINUE
      DO 56 I= 1,N
   56 TORDER(I) = I
      TPOINT = N1
      MOST = MOST + 1
   50 MOST = MOST - 1
      IF(MOST.LT.0)GO TO 54
      DO 55 I=1,NR
      II = GORDER(I)
      IF(II.LE.N)GO TO 55
      IF(USED(II).NE.MOST)GO TO 55
      TORDER(TPOINT) = II
      TPOINT = TPOINT + 1
   55 CONTINUE
      GO TO 50
   54 CONTINUE
C     INITIALIZE GSMALL
      DO 4 I=N1,NM
      X = (I-1)*N2
      DO 4 J=1,N2
      Y = P$(1,X+J)
      IF(Y.EQ.0)GSMALL(I,J) = -100
      IF(Y.EQ.1)GSMALL(I,J) = 1
      IF(Y.EQ.-1)GSMALL(I,J)=0
    4 CONTINUE
      EFLAG = 0
      GO TO 57
      ENTRY INITGS
      EFLAG = 1
   57 DO 3 I=1,NR
      USED(I) = 0
      IF(I.LT.N1)GO TO 58
      IF(I.GT.NM) GO TO 58
      GO TO 3
   58 DO 59 J = 1,N2
   59 GSMALL(I,J)= 0
    3 CONTINUE
      DO 62 I = N1,NM
      USED(I) = 1
   62 CONTINUE
C     INITIALIZATION
      DO 34 I=1,NR
      GATE = GORDER(I)
      IF(GATE.LT.N1)GO TO 34
      XX= LIPRED(GATE)
      IF(XX.EQ.0)GOTO34
      F$UP1 = 0
      F$UB0 = 0
      DO 35 J=1,N2
      COMPNT = GSMALL(GATE,J)
      IF(COMPNT.EQ.0)GO TO 35
      IF(COMPNT.LT.0)GO TO 36
      IF(COMPNT.GE.1000) GO TO 35
      F$UB0 = F$UB0 + 1
      F$0(F$UB0) = J
      GO TO 35
   36 IF(COMPNT.LE.-1000) GO TO 35
      F$UB1 = F$UB1 + 1
      F$1(F$UB1) = J
   35 CONTINUE
      IF(F$UB1.EQ.0)GO TO 34
      DO 38 K=1,XX
      FEEDGT = IPRED(K,GATE)
      X = (FEEDGT-1)*N2
      DO 39 L=1,F$UB1
      Y = F$1(L)
      IF(P$(1,X+Y).LE.0)GO TO 39
      IF(GSMALL(FEEDGT,Y).GT.1000)GOTO39
      IF(GSMALL(GATE,Y).EQ.-200)GOTO39
      IF(GSMALL(GATE,Y).EQ.-100)GO TO 40
      GSMALL(GATE,Y) = -200
      GO TO 39
   40 GSMALL(GATE,Y) = -FEEDGT
   39 CONTINUE
   38 CONTINUE
      DO 60 K=1,XX
   60 MARKED(IPRED(K,GATE)) = 0
      DO 41 K=1,F$UB1
      X = GSMALL(GATE,F$1(K))
      IF(X.EQ.-100)GO TO 41
      IF(X.EQ.-200)GOTO41
      X = -X
      GSMALL(+X,F$1(K))=1
      USED(X) = 1
      IF(MARKED(X).EQ.1)GOTO41
      MARKED(X) = 1
      DO 42 L=1,F$UB0
      Y = GSMALL(X,F$0(L))
      IF(Y.GT.1000.OR.Y.LT.-1000)GO TO 42
      GSMALL(+X,F$0(L))=-100
   42 CONTINUE
   41 CONTINUE
   34 CONTINUE
      IF(FFLAG.EQ.1)RETURN
C     INITIALIZE COUNTER TO LOOP ONCE FOR EACH GATE
      GCOUNT = 0
C     INCREMENT GCOUNT
    5 GCOUNT = GCOUNT + 1
C     ARE ALL GATES EXHAUSTED?
      IF(GCOUNT.LE.NR)GO TO 6
      IF(T.GT.0) IMPROV = 1
      IF(IMPROV.EQ.0)RETURN
C     IF HERE, NETWORK WAS ALTERED, SO UPDATE ARRAYS
      CALL SUBNET
      CALL PVALUE
      RETURN
    6 GCD = GORDER(GCOUNT)
C     IS GCD AN ISOLATED GATE OR EXTERNAL VARIABLE?
      IF(GCD.LE.N)GOTO5
      DO 8 I=1,N2
      IF(GSMALL(GCD,I).GE.1)GOTO7
    8 CONTINUE
C     IF HERE, GATE IS ISOLATED - REMOVE INPUTS
      X = LIPRED(GCD)
      IF(X.EQ.0)GOTO5
      DO 9 I=1,X
      Y = IPRED(I,GCO)
      INC$MX(Y,GCO) = 0
C     RECORD THE DISCONNECTION
      T = T + 1
    9 CONTINUE
      GOTO 5
C     REMOVE UNNECESSARY CONNECTIONS TO GCO IN THE NEXT FEW SECTIONS
C
C     CALCULATE F(GCO)
    7 F$UB1 = 0
      DO 10 I=1,N2
      IF(GSMALL(GCO,I).GE.0)GOTO10
      F$UB1 = F$UB1 + 1
      F$1(F$UB1) = I
   10 CONTINUE
      DO 11 I=1,F$UB1
   11 INPTCV(F$1(I)) = 0
      X = LIPRED(GCO)
      DO 222I=1,X
      ESS1S(IPRED(I,GCO)) = 0
  222 CONTINUE
      T1SUB = 0
      T2SUB = 0
      DO 48 I = 1,NR
      IF(INC$MX(I,GCO).EQ.0)GOTO48
      T1SUB = T1SUB + 1
      T1PRED(T1SUB) = I
   48 CONTINUE
   17 DO 18 I=1,X
      Y = (T1PRED(I)-1)*N2
      DO 19 J=1,F$UB1
      Q = F$1(J)
      IF(P$(1,Y+Q).NE.1)GO TO 19
      IF(INPTCV(Q).LE.0) GO TO 20
      INPTCV(Q) = INPTCV(Q) + 1
      GO TO 19
   20 IF(INPTCV(Q).LT.0)GO TO 21
      INPTCV(Q) = -T1PRED(I)
      GO TO 19
   21 INPTCV(Q) = 2
   19 CONTINUE
   18 CONTINUE
C     MARK ESSENTIAL 1'S
      DO 22 I=1,F$UB1
      Q = INPTCV(F$1(I))
      IF(Q.GE.0)GO TO 22
      ESS1S(-Q) = ESS1S(-Q) + 1
   22 CONTINUE
   46 SELECT = 0
      BESTSL = 0
      DO 45 L=1,X
      Q = T1PRED(L)
      IF(INC$MX(Q,GCO).EQ.0)GOTO45
      IF(ESS1S(Q).GT.0)GOTO45
      IF(SELECT.EQ.0)SELECT = Q
      IF(USED(Q).EQ.1)GOTO45
      IF(BESTSL.NE.0)GOTO45
      BESTSL = Q
   45 CONTINUE
      IF(SELECT.EQ.0)GO TO 47
      Q = SELECT
      IF(BESTSL.NE.0)Q = BESTSL
C     IF HERE, GATE HAS NO ESSENTIAL 1'S  -  REMOVE IT
      INC$MX(Q,GCO) = 0
      T = T + 1
C     UPDATE ESS1S
      Y = (Q - 1)*N2
      DO 24 J=1,F$UB1
      V = F$1(J)
      IF(P$(1,Y+V).NE.1)GO TO 24
C     UPDATE INPTCV FOR COMPONENT V
      INPTCV(V) = INPTCV(V) - 1
      IF(INPTCV(V).GT.1)GO TO 24
C     CASE WHEN NEW ESSEN 1 CREATED
      DO 27 K = 1,X
      W = T1PRED(K)
      IF(INC$MX(W,GCO).EQ.0) GO TO 27
      Z = (W - 1) * N2
      IF(P$(1,Z+V).EQ.0)GO TO 27
      ESS1S(W) = ESS1S(W) + 1
C     IN THIS CASE, NO NEED TO UPDATE INPTCV FURTHER
      GSMALL(GCO,V) = -W
      GO TO 24
   27 CONTINUE
   24 CONTINUE
      GOTO46
   47 DO 49 I = 1,NR
      IF(INC$MX(I,GCO).EQ.0)GOTO49
      T2SUB = T2SUB + 1
      T2PRED(T2SUB) = I
   49 CONTINUE
C     NOW ALL CURRENT INPUTS HAVE ESSENTIAL 1'S
C     INPUTS STILL CONNECTED TO GCO ARE LISTED IN T2PRED IN REVERSE
C     ORDER
C
C     UPDATE G(I)'S OF THOSE GATES STILL CONNECTED TO GATE GCO
C
      DO 29 II=1,F$UB1
      I = F$1(II)
      CHOICE = -GSMALL(GCO,I)
      IF(CHOICE.LT.100)GO TO 61
      CHOICE = 0
      DO 30 JJJ=1,NR
      JJ = TORDER(JJJ)
      IF(INC$MX(JJ,GCO).EQ.0)GO TO 30
      IF(P$(1,(JJ-1)*N2+I).NE.1)GO TO 30
      IF(JJ.LE.N)GO TO 29
      IF(CHOICE.EQ.0)CHOICE=JJ
      IF(GSMALL(JJ,I).GE.1)GOTO29
   30 CONTINUE
   61 GSMALL(CHOICE,I) = 1
      USED(CHOICE) = 1
   29 CONTINUE
      DO 32 I=1,N2
      IF(GSMALL(GCO,I).LT.1)GO TO 32
      DO 33 J=1,T2SUB
      IF(GSMALL(T2PRED(J),I).EQ.0)GSMALL(T2PRED(J),I)=-100
   33 CONTINUE
   32 CONTINUE
      GOTO5
      END
      SUBROUTINE ORDRQ2                                                 E1 08220
C**** THIS SUBROUTINE MAKES A LIST OF PREDECESSORS OF GI                 E1 08230
C     ACCORDING TO ORDERING Q2                                          E1 08240
C     LIP IS STORED AT COMMON STORAGE                               ****E1 08250
C                                                                       E1 08260
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.   E1 08270
C                                                                       E1 08280
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                            E1 08290
      COMMON NEPMAX                                                     E1 08300
      COMMON    N             , M             , A             , B       E1 08310
     1     ,    R             , N2            , N1            , NR      E1 08320
     2     ,    NM            , KFLAG         , JFLAG         , COST    E1 08330
     3     ,    LEVM          , NRN2          , NM1           , NN2     E1 08340
      COMMON    ISUCC(40,40)  , LISUCC(40)    , IPRED(40,40)  , LIPRED(40)  E1 08350
     1     ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)   E1 08360
     2     ,    GLEVEL(40)    , LGLIST(40)    , HLIST(40,40)  , TIME    E1 08370
      COMMON    T             , RTCONN(100)   , S             , RSCONN(100) E1 08380
      COMMON    IFLAG         ,POINTA         ,ESS1S(40)      ,F$1(32)  E1 08390
     1     ,F$UB1             ,INPTCV(32)     ,LISTC(40)      ,POINTC   E1 08400
     2     ,LISTL(40)         ,POINTL         ,ORIGIN(40)     ,IPATH(40)    E1 08410
     3     ,POINTR            ,VF$1(32)       ,VF$UB1         ,GSMALL(40,32)E1 08420
      COMMON    POTAB(200,42),PPOTAB(40)      ,LPOTAB(40)     ,NRPLC(2) E1 08430
     1     ,RPLC(2,40)        ,IDX0(32)       ,IDX0E(32)      ,IDX1(32) E1 08440
     2     ,IDX1E(32)         ,SUMP(32)       ,SETT1(32)      ,NOT1     E1 08450
     3     ,SETS1(40)         ,NOS1           ,SETS(40)       ,NOS      E1 08460
     4     ,STS               ,SUMS2(32)      ,SETS2(200)     ,NOS2     E1 08470
     5     ,LIP               ,NOOE           ,KEYA           ,KEYB     E1 08480
     6     ,NO0               ,NO1            ,NO1E           ,$GT      E1 08490
     7     ,$LTH              ,$PW            ,$NOE           ,GI       E1 08500
      COMMON                  NOT1SV          ,NOS1SV         ,LMTS2    E1 08510
      DIMENSION  WRPLC(2,40)                                            E1 08520
      NRPLC(1) = 0                                                      E1 08530
      NRPLC(2) = 0                                                      E1 08540
      DO 8100 LI=1,LIP                                                  E1 08550
        GP=IPRED(LI,GI)                                                 E1 08560
        IF(GP.LE.N) GO TO 8100                                          E1 08570
        IF(INC$MX(GP,GI).EQ.0) GO TO 8100                               E1 08580
        BSGP=(GP-1)*N2                                                  E1 08590
      NOONEE = 0                                                        E1 08600
        ESSN=1                                                          E1 08610
        DO 8050 NO=1,NOOE                                               E1 08620
          TH=IDX0E(NO)                                                  E1 08630
          IF(P$(1,BSGP+TH).LE.0) GO TO 8050                             E1 08640
          NOONEE=NOONEE+1                                               E1 08650
          IF(SUMP(TH).EQ.1) ESSN=2                                      E1 08660
 8050   CONTINUE                                                        E1 08670
C**** PUT GP INTO RPLC(1,*) OR RPLC(2,*) DEPENDING ON ESSN               E1 08680
C                       ESSN=1 :  NO ESSENTIAL ERROR                    E1 08690
C                       ESSN=2 :  WITH ESSENTIAL ERRORS                 E1 08700
C     RPLC TABLES ARE STORED ACCORDING TO ORDERING Q1(NOONEE)           E1 08710
C                       NRPLC(ESSN) :NUMBER OF ELEMENTS IN RPLC(ESSN)**E1 08720
      IF(NOONEE.EQ.0) GO TO 8100                                        E1 08730
      P2=NRPLC(ESSN)                                                    E1 08740
      IF(P2.EQ.0) GO TO 8070                                            E1 08750
       DO 8060 PR=1,P2                                                  E1 08760
        RP=P2-PR+1                                                      E1 08770
        IF(WRPLC(ESSN,RP).LE.NOONEE) GO TO 8080                         E1 08780
        WRPLC(ESSN,RP+1)=WRPLC(ESSN,RP)                                 E1 08790
        RPLC(ESSN,RP+1)=RPLC(ESSN,RP)                                   E1 08800
 8060  CONTINUE                                                          E1 08810
 8070  RP=0                                                              E1 08820
 8080  WRPLC(ESSN,RP+1)=NOONEF                                           E1 08830
       RPLC(ESSN,RP+1)=GP                                                E1 08840
       NRPLC(ESSN)=NRPLC(ESSN)+1                                         E1 08850
 8100  CONTINUE                                                          E1 08860
       RETURN                                                            E1 08870
       END                                                               E1 08880


       SUBROUTINE OUTPUT(MATRIX,ARRAY)                                   E1 08890
C                                                                        E1 08900
C      DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.  E1 08910
C                                                                        E1 08920
       IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                            E1 08930
       COMMON NERMAX                                                     E1 08940
       COMMON    N               , M              , A              , B   E1 08950
     1      ,    R               , N2             , N1             , NR  E1 08960
     2      ,    NM              , KFLAG          , JFLAG          , COST E1 08970
     3      ,    LEVM            , NRN2           , NM1            , NN2 E1 08980
       COMMON    ISUCC(40,40) , LISUCC(40)    , IPRED(40,40) , LIPRED(40) E1 08990
     1      ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)  E1 09000
     2      ,    GLEVEL(40)   , LGLIST(40)    , HLIST(40,40) , TIME       E1 09010
       COMMON    T               , RTCONN(100)  , S              , RSCONN(100) E1 09020
       COMMON    IFLAG           ,POINTA          ,ESS1S(40)      ,F$1(32)  E1 09030
     1      ,F$UB1               ,INPTCV(32)      ,LISTC(40)      ,POINTC   E1 09040
     2      ,LISTL(40)           ,POINTL          ,ORIGIN(40)     ,IPATH(40) E1 09050
     3      ,POINTR              ,VF$1(32)        ,VF$UB1         ,GSMALL(40,32)E1 09060
       COMMON    POTAB(200,42),PPOTAB(40)         ,LPOTAB(40)     ,NRPLC(2)  E1 09070
     1      ,RPLC(2,40)          ,IDX0(32)        ,IDX0E(32)      ,IDX1(32)  E1 09080
     2      ,IDX1E(32)           ,SUMP(32)        ,SETT1(32)      ,NOT1      E1 09090
     3      ,SETS1(40)           ,NOS1            ,SETS(40)       ,NOS       E1 09100
     4      ,STS                 ,SUMS2(32)       ,SETS2(200)     ,NOS2      E1 09110
     5      ,LIP                 ,NOOF            ,KEYA           ,KEYB      E1 09120
     6      ,NOO                 ,NO1             ,NO1E           ,$GT       E1 09130
     7      ,$LTH                ,$PW             ,$NOE           ,GI        E1 09140
       COMMON                    NOT1SV         ,NOS1SV         ,LMTS2       E1 09150
       DIMENSION UX(5), UY(5), UG(40), UF(40), ARRAY(40), ARRAY2(2,1280) E1 09160
       DIMENSION MATRIX(40,40)                                           E1 09170
       DATA UX /' X1',' X2',' X3',' X4',' X5'/                           E1 09180
       DATA UY/' Y1',' Y2',' Y3',' Y4',' Y5'/                            E1 09190
       DATA UF /'  1','  2','  3','  4','  5','  6','  7','  8'          E1 09200
     1        ,  '  9',' 10',' 11',' 12',' 13',' 14',' 15',' 16'         E1 09210
     2        ,  ' 17',' 18',' 19',' 20',' 21',' 22',' 23',' 24'         E1 09220
     3        ,  ' 25',' 26',' 27',' 28',' 29',' 30',' 31',' 32'         E1 09230
     4        ,  ' 33',' 34',' 35',' 36',' 37',' 38',' 39',' 40'/        E1 09240
       DATA GMAX/40/                                                     E1 09250
C                                                                        E1 09260
       KEYXC=ARRAY(1)                                                    E1 09270
       IF(KEYXC.NE.0) GO TO 50                                           E1 09280
       DO 1 GI=1,N                                                       E1 09290
        UNAME(GI)=UX(GI)                                                 E1 09300
    1  CONTINUE                                                          E1 09310
       GO TO 100                                                         E1 09320
   50  CONTINUE                                                          E1 09330
       L=N/2                                                             E1 09340
       DO 4 GI=1,L                                                       E1 09350
        UNAME(GI)=UX(GI)                                                 E1 09360
       UNAME(GI+L)=UY(GI)                                                E1 09370
    4  CONTINUE                                                          E1 09380
  100  CONTINUE                                                          E1 09390
```

```
      DO 2 GI=N1,GMAX
       UNAME(GI)=UF(GI-N)
    2 CONTINUE
      RETURN
C
      ENTRY LINE(L)
      DO 6 LL=1,L
       PRINT 5
    5 FORMAT(1H )
    6 CONTINUE
      RETURN
C
      ENTRY PAGE
      PRINT 7
    7 FORMAT(1H1)
      RETURN
C
      ENTRY CKT(MATRIX,ARRAY)
      PRINT 10
   10 FORMAT(1H ,8X,'GATE .. LEVEL',6X, 'FED BY'/)
      DO 20 GJ=N1,NR
       G=0
       DO 15 GI=1,NR
        IF(MATRIX(GI,GJ).EQ.0) GO TO 15
        G=G+1
        UG(G)=UNAME(GI)
   15  CONTINUE
       IF(G.EQ.0) GO TO 18
       PRINT 17, UNAME(GJ),ARRAY(GJ),(UG(GG),GG=1,G)
   17  FORMAT(1H0, 9X,A3,5X,'/',I2,'/',5X,35(    A3))
       GO TO 20
   18  PRINT 19, UNAME(GJ),ARRAY(GJ)
   19  FORMAT(1H0, 9X,A3,5X,'/',I2,'/')
   20 CONTINUE
      RETURN
C
      ENTRY TRUTH(ARRAY2,J)
      IF(J.EQ.2) GO TO 36
      PRINT 35
   35 FORMAT(11X,    'TRUTH TABLE'/)
      GO TO 38
   36 PRINT 37
   37 FORMAT(11X,  'REQUIREMENT TABLE')
   38 CONTINUE
      DO 40 GI=1,NR
       ILO=(GI-1)*N2+1
       IHI=ILO+N2-1
       PRINT 41, UNAME(GI), (ARRAY2(J,I),I=ILO,IHI)
   40 CONTINUE
   41 FORMAT(1H0, 9X,A3,' = ', 32(I1,1X))
      RETURN
      END


      SUBROUTINE POT
C
C     THIS SUBROUTINE GENERATES THE POTENTIAL OUTPUT TABLE
C
C     POTAB(P1,P2) STORES POTENTIAL OUTPUT TABLE
C          P1=1,$MXPTR: FUNCTION ENTRY NUMBER
C          P2=1,32   : VALUE OF EACH COMPONENT OF THAT FUNCTION
```

```
C          P2=33($GT) : GATE NUMBER WHERE THE FUNCTION IS REALIZED
C          P2=34($LTH): NUMBER OF CONNECTIONS TO BE ADDED
C          P2=35,40   : GATES WHICH ARE TO BE CONNECTED TO POTAB(*,$GT)
C          P2=41($PW) : PREFERENCE WEIGHT
C          P2=42($NOE): NUMBER OF ONE ERRORS
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
      COMMON NERMAX
      COMMON    N              , M              , A              , B
     1     ,    R              , N2             , N1             , NP
     2     ,    NM             , KFLAG          , JFLAG          , COST
     3     ,    LEVM           , NPN2           , NM1            , NN2
      COMMON    ISUCC(40,40)   , LISUCC(40)     , IPRED(40,40)   , LIPRED(40)
     1     ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)
     2     ,    GLEVEL(40)     , LGLIST(40)     , HLIST(40,40)   , TIME
      COMMON    T              , RTCONN(100)    , S              , RSCONN(100)
      COMMON    IFLAG          ,POINTA          ,ESS1S(40)       ,F$1(32)
     1     ,F$UB1              ,INPTCV(32)      ,LISTC(40)       ,POINTC
     2     ,LISTL(40)          ,POINTL          ,ORIGIN(40)      ,IPATH(40)
     3     ,PO$NTR             ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)        ,LPOTAB(40)      ,NRPLC(2)
     1     ,RPLC(2,40)         ,IDX0(32)        ,IDX0E(32)       ,IDX1(32)
     2     ,IDX1E(32)          ,SUMP(32)        ,SETT1(32)       ,NOT1
     3     ,SETS1(40)          ,NOS1            ,SETS(40)        ,NOS
     4     ,STS                ,SUMS2(32)       ,SETS2(200)      ,NOS2
     5     ,LIP                ,NOOF            ,KEYA            ,KEYB
     6     ,NO0                ,NO1             ,NO1E            ,$GT
     7     ,$LTH               ,$PW             ,$NOE            ,GI
      COMMON                   NOT1SV          ,NOS1SV          ,LMTS2
      DIMENSION  INDEX(32)
      DATA $MXPTR/200/
C**** INITIALIZE PPOTAB(*) ****
      DO 90 GI=1,NP
   90 PPOTAB(GI)=0
      POINTR = 1
      DO 980 LEVV=1,LEVM
        LEV=LEVM-LEVV+1
        LGL =LGLIST(LEV)
        DO 960 LG=1,LGL
          GI = HLIST(LG,LEV)
          IF(LEV.GT.1) GO TO 100
          IF(GI.GT.NM.OR.GI.LE.N.OR.M.EQ.1) GO TO 960
  100     LISI = LISUCC(GI)
          BSGI = (GI-1)*N2
          IF(POINTR.GT.$MXPTR) GO TO 990
          PPOTAB(GI) = POINTR
C****     COPY PRESENT OUTPUT
          DO 110 TH=1,N2
            POTAB(POINTR,TH)=P$(1,BSGI+TH)
  110     CONTINUE
          POTAB(POINTR,$GT)=GI
          POTAB(POINTR,$LTH)=0
          POINTR = POINTR + 1
          IF(GI.LE.NM) GO TO 950
          DO 380 LEVJ=LEV,LEVM
            LGLJ=LGLIST(LEVJ)
            DO 360 LGJ=1,LGLJ
               GJ=HLIST(LGJ,LEVJ)
               IF(INC$MX(GJ,GI).GT.0.OR.GI.EQ.GJ) GO TO 360
```

```
C**** CHECK IF GJ IS CONNECTED TO ALL SUCCESSORS OF GI                 ****
               DO 120 LII=1,LISI
                 IF(INC$MX(GJ,ISUCC(LII,GI)).LE.0) GO TO 360
   120         CONTINUE
C**** CHECK IF GJ IS STRONGLY CONNECTIBLE TO GI                         ****
               BSGJ = (GJ-1)*N2
               NO = 0
               DO 180 TH = 1,N2
                 IF(P$(1,BSGI+TH).NE.1.OR.P$(1,BSGJ+TH).NE.1)GO TO 180
                 NO=NO+1
                 INDEX(NO)=TH
   180         CONTINUE
C****          NO=0   =>  NOT STRONGLY CONNECTABLE                     ****
C****          NO>0   =>  STRONGLY CONNECTABLE                         ****
               IF(NO.EQ.0) GO TO 360
              IF(POINTR.GT.$MXPTR) GO TO 990
               DO 200 TH=1,N2
                 POTAB(POINTR,TH)=P$(1,BSGI+TH)
   200         CONTINUE
               DO 210 NORUN=1,NO
                 POTAB(POINTR,INDEX(NORUN))=0
   210         CONTINUE
               SP=PPOTAB(GI)+1
               IF(POINTR.EQ.SP) GO TO 300
               EP = POINTR - 1
C**** CHECK IF THIS ENTRY IS SAME AS ONE OF THE PREVIOUS ENTRIES        ****
               DO 230 PTR=SP,EP
                   DO 220 TH=1,N2
                     IF(POTAB(POINTR,TH).NE.POTAB(PTR,TH)) GO TO 230
   220             CONTINUE
                   GO TO 360
   230         CONTINUE
   300         POTAB(POINTR,$GT)=GI
               POTAB(POINTR,$LTH)=1
               POTAB(POINTR,$LTH+1)=GJ
               POINTR = POINTR + 1
   360       CONTINUE
   380     CONTINUE
C****     IF THE SET OF STRONGLY CONNECTABLE GATES CONTAINS MORE
C         THAN ONE GATE TRY THEIR COMBINATIONS                         ****
C              PTR1: STARTING POSITION OF THE LIST
C              PTR2: STARTING POSITION OF THE COMBINATIONS OF THE LIST
C              PTR : THE ENTRY WHOSE COMBINATIONS WITH OTHERS ARE UNDER
C                    CONSIDERATION
C              PTRL: LAST ENTRY PRECEDING PTR
C              PTRL2:LAST ENTRY OF COMBINATIONS OF ENTRIES PRECEDING PTR
C
          PTR1=PPOTAB(GI)+1
          PTR2=POINTR
          EP=PTR2-1
          SP=PTR1+1
          IF(SP.GT.EP) GO TO 950
          IF(PTR2-PTR1.GT.6) PRINT 1110
 1110 FORMAT(/////21X,'***** WARNING: NUMBER OF STRONGLY CONNECTABLE FUN
    1CTIONS FOR A GATE EXCEEDS 6 *****'/
     2            21X,'***** NOT ALL POSSIBLE OUTPUTS ARE AVAILABLE IN C
     3ALCULATION                 *****'////)
          DO 560 PTR=SP,EP
            PTRL=PTR-1
            PTRL2=POINTR-1
C****       MAKE THE COMBINATIONS OF PTR AND ENTRIES PRECEDING IT AS
```

```
C           TH NEW ENTRIES                                          ****
            DO 450 PT=PTR1,PTRL
C****       MAKE NEW ENTRY AS THE COMBINATION OF ENTRIES PTR AND PT ****
            IF(POINTR.GT.$MXPTR) GO TO 990
               DO 420 TH=1,N2
                 POTAB(POINTR,TH)=1
                 IF(POTAB(PTR,TH).EQ.0.OR.POTAB(PT,TH).EQ.0)
     1           POTAB(POINTR,TH)=0
  420          CONTINUE
            POTAB(POINTR,$GT)=GI
            POTAB(POINTR,$LTH)=2
            POTAB(POINTR,$LTH+1)=POTAB(PT,$LTH+1)
            POTAB(POINTR,$LTH+2)=POTAB(PTR,$LTH+1)
            POINTR=POINTR+1
  450       CONTINUE
            IF(PTR2.GT.PTRL2) GO TO 560
C****       MAKE THE COMBINATIONS OF PTR AND THE COMBINATIONS OF ENTRIES
C           PRECEDING PTR AS THE NEW ENTRIES                        ****
            DO 520 PT=PTR2,PTRL2
            IF(POINTR.GT.$MXPTR) GO TO 990
            IF(POTAB(PT,$LTH).GE.6) GO TO 520
               DO 480 TH=1,N2
                 POTAB(POINTR,TH)=1
                 IF(POTAB(PTR,TH).EQ.0.OR.POTAB(PT,TH).EQ.0)
     7           POTAB(POINTR,TH)=0
  480          CONTINUE
               LTH=POTAB(PT,$LTH)+2
               DO 500 TH=1,LTH
               POTAB(POINTR,$GT+TH-1)=POTAB(PT,$GT+TH-1)
  500          CONTINUE
               POTAB(POINTR,$LTH)=POTAB(POINTR,$LTH)+1
               POTAB(POINTR,$GT+LTH)=POTAB(PTR,$LTH+1)
               POINTR=POINTR+1
  520       CONTINUE
  560     CONTINUE
  950     LPOTAB(GI)=POINTR-1
  960   CONTINUE
  980 CONTINUE
      RETURN
C**** NUMBER OF POSSIBLE OUTPUT TABLE ENTRIES EXCEEDS THE LIMIT *****
  990 CALL LINE(5)
      PRINT 1000, $MXPTR
 1000 FORMAT(21X,'**** WARNING: NUMBER OF POSSIBLE OUTPUT TABLE ENTRIES
     1EXCEEDS THE LIMIT($MXPTR=',I3,') ****'/
     2       21X,'**** NOT ALL POSSIBLE OUTPUTS ARE AVAILABLE IN CALCULA
     3TION                         ****'///)
      LPOTAB(GI) = $MXPTR
      RETURN
      END


      SUBROUTINE PROCCE(WORKED)
C     EDITION AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
C     IF PROCCE SUCCESSFULLY COMPENSATES ERRORS, 'WORKED' IS SET TO 1, O
C     'WORKED' IS SET TO 0
C
C     DEFS. OF MOST  'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
C     VARIABLE DEFINITIONS:
C         EP: EP(I)=1 MEANS AT LEAST ONE NETWORK OUTPUT GATE HAS AN
C             ERRONEOUS OUTPUT IN THE I-TH COMPONENT WHEN PCO IS REMOVED
```

```
C               FROM THE NETWORK.  FP(I)=0 OTHERWISE.                          E1 11800
C       ERRORS: TOTAL NO. OF ERRORS IN NETWORK OUTPUTS WHEN PCO REMOVED.  E1 11810
C        GATES: NUMBER OF GATES REMOVED FROM NETWORK BY CALL TO MINI2.     E1 11820
C       IMPROV: A PARAMETER RETURNED BY MINI2.  '=1' MEANS MINI2 WAS ABLE E1 11830
C               TO REDUCE COST OF NETWORK.                                 E1 11840
C          MAX: MAXIMUM NUMBER OF REQUIRED  1'S IN A CSPF VECTOR (AFTER    E1 11850
C               CALLING MINI2) PLUS 1.                                     E1 11860
C          MIN: ORIGINALLY SET TO ZERO, MIN IS INCREMENTED EACH TIME BY 1 E1 11870
C               UNTIL ITS VALUE EQUALS MAX.                                E1 11880
C          NEP: NO. OF ERROR POSITIONS FOR A GIVEN NETWORK AFTER A SE-     E1 11890
C               LECTED GATE HAS BEEN REMOVED.  AN ERROR POSITION IS A      E1 11900
C               COMPONENT POSITION WHICH IS IN ERROR FOR AT LEAST ONE      E1 11910
C               OUTPUT.                                                    E1 11920
C       NEPMAX: READ FROM INPUT CARDS, THIS PARAMETER IS PASSED TO PROCCE E1 11930
C               WHEN IT IS CALLED BY MAIN.  IT REPRESENTS THE MAXIMUM      E1 11940
C               ALLOWABLE NUMBER OF ERROR POSITIONS.  IF AN ALTERED (I.E.,E1 11950
C               SOME PCO REMOVED) NETWORK EXCEEDS THIS MAXIMUM, ERROR      E1 11960
C               COMPENSATION IS NOT ATTEMPTED FOR THAT NETWORK.            E1 11970
C       NETOUT: STORES OUTPUTS OF GATES IN ALTERED (PCO REMOVED) NETWORK. E1 11980
C       ONECNT: USED IN COUNTING NO. OF 1'S IN CSPF VECTOR OF A GATE.      E1 11990
C         ONES: AFTER THE INITIAL CALCULATION OF THE CSPF SETS IN THE      E1 12000
C               BEGINNING, ONES(GI) GIVES THE NUMBER OF 1'S IN THE CSPF    E1 12010
C               VECTOR OF GI.  THIS INFORMATION IS REQUIRED FOR GENERATINGE1 12020
C               PORDER.                                                    E1 12030
C       ORGOUT: USED TO STORE ORIGINAL (UNALTERED) NETWORK OUTPUTS IN      E1 12040
C               CODED FORM (SAME CODE AS IN GSMALL) AND (40,32) FORMAT.    E1 12050
C          PCO: CURRENT GATE REMOVED FROM ORIGINAL NETWORK TO OBTAIN       E1 12060
C               CURRENT ALTERED NETWORK.  PCO = PORDER(PCOUNT).            E1 12070
C       PCOUNT: A POINTER TO PORDER.                                       E1 12080
C       PORDER: ORDERING OF GATES ACCORDING TO NUMBER OF 1'S IN THEIR      E1 12090
C               CSPF VECTORS.  GATES ARE INDIVIDUALLY REMOVED FROM ORIGI- E1 12100
C               NAL NETWORK IN THIS ORDER                                  E1 12110
C         PSUB: USED AS A POINTER TO PORDER DURING ITS INITIALIZATION.     E1 12120
C       QINC$M: STORES A COPY OF INC$MX FOR THE ORIGINAL NETWORK.          E1 12130
C        START: POINTS TO BEGINNING OF LIST OF NETWORK OUTPUTS IN P$.      E1 12140
C         STOP: POINTS TO END OF LIST OF NETWORK OUTPUTS IN P$.            E1 12150
C                                                                          E1 12160
C       I,J,NI,X,Y ARE USED AS JUST TEMPORARY VARIABLES.                   E1 12170
C                                                                          E1 12180
C       HOW TO INCREASE CAPACITY OF SUBROUTINE.                            E1 12190
C       DIMENSION: PORDER(X)                                               E1 12200
C                  ONES(X)                                                 E1 12210
C                  QINC$M(X,X) - X EQUAL TO MAX NO. OF GATES PLUS EX. VAR.E1 12220
C                  FP(Y)       - Y EQUAL TO: 2**(MAX ALLOWED NO OF EX VAR)E1 12230
C                  NETOUT(X,Y)                                             E1 12240
C                  ORGOUT(X,Y) - X,Y AS ABOVE                              E1 12250
C                                                                          E1 12260
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                               E1 12270
      COMMON NEPMAX                                                        E1 12280
      COMMON    N              , M              , A              , B       E1 12290
     1     ,    R              , N2             , N1             , NR      E1 12300
     2     ,    NM             , KFLAG          , JFLAG          , COST    E1 12310
     3     ,    LEVM           , NRN2           , NM1            , NN2     E1 12320
      COMMON    ISUCC(40,40) , LISUCC(40)     , IPRED(40,40) , LIPRED(40)  E1 12330
     1     ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)     , UNAME(40)   E1 12340
     2     ,    GLEVEL(40)     , LGLIST(40)     , HLIST(40,40) , TIME      E1 12350
      COMMON    T              , RTCONN(100)    , S              , RSCONN(100) E1 12360
      COMMON    IFLAG          ,POINTA          ,ESS1S(40)       ,F$1(32)  E1 12370
     1      ,F$UB1             ,INPTCV(32)      ,LISTC(40)       ,POINTC   E1 12380
     2      ,LISTL(40)         ,POINTL          ,ORIGIN(40)      ,IPATH(40) E1 12390
     3      ,POINTR            ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)E1 12400
```

```
      COMMON    POTAB(200,42),PPOTAB(40)     ,LPOTAB(40)     ,NRPLC(2)
     1      ,RPLC(2,40)      ,IDX0(32)       ,IDX0E(32)      ,IDX1(32)
     2      ,IDX1E(32)       ,SUMP(32)       ,SETT1(32)      ,NOT1
     3      ,SETS1(40)       ,NOS1           ,SETS(40)       ,NOS
     4      ,STS             ,SUMS2(32)      ,SETS2(200)     ,NOS2
     5      ,LIP             ,NO0E           ,KEYA           ,KEYB
     6      ,NO0             ,NO1            ,NO1E           ,$GT
     7      ,$LTH            ,$PW            ,$NOE           ,GI
      COMMON                 NOT1SV         ,NOS1SV         ,LMTS2
      DIMENSION PORDER(40),ONES(40),QINC$M(40,40),NETOUT(40,32),
     1 EP(32),ORGOUT(40,32)
C     THIS SUBROUTINE ASSUMES ALL ARRAYS ARE UPDATED
C     PREVIOUS TO BEING CALLED
C
      $GT  = 33
      $LTH = 34
      $PW  = 41
      $NOE = 42
      WORKED = 0
      S = 0
      T = 0
C
C     BLOCK  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B
C
      CALL MINI2(IMPROV)
C     IN THIS CALL TO MINI2, GORDER WILL BE CALCULATED.  GORDER WILL BE
C     LATER IN EACH CALL TO INITGS (AN ENTRY POINT OF MINI2). NOTE THAT
C     IS NOT AFFECTED BY THE REMOVAL OF GATES FROM THE ORIGINAL NETWORK.
      IF(IMPROV.EQ.0)GO TO 1
      GATES = 0
      DO 2 I = NM1,NR
      DO 3 J = N1,NR
      IF(INC$MX(I,J).GT.0)GO TO 2
    3 CONTINUE
      GATES = GATES + 1
    2 CONTINUE
      PRINT 4,GATES,T
    4 FORMAT(' ',I5,' GATES AND',I3,' CONNECTIONS HAVE BEEN REMOVED FROM
     1 THE NETWORK DURING THE INITIAL CALCULATION OF THE CSPF SET')
    1 CONTINUE
C     COUNT THE NUMBER OF 1'S IN THE CSPF VECTOR FOR EACH GATE
      MAX = 0
      DO 5 I = N1,NR
      ONECNT = 0
      DO 6 J = 1,N2
      IF(GSMALL(I,J).LE.0)GO TO 6
      ONECNT = ONECNT + 1
    6 CONTINUE
      IF(ONECNT.GT.MAX)MAX=ONECNT
      ONES(I) = ONECNT
    5 CONTINUE
      MAX = MAX + 1
      MIN = -1
      PSUB = 1
    7 MIN = MIN + 1
      IF(MIN.EQ.MAX) GO TO 8
      DO 9 I = N1,NR
      IF(ONES(I).NE.MIN)GO TO 9
      PORDER(PSUB) = I
      PSUB = PSUB + 1
    9 CONTINUE
```

```
      GOTO7                                                             E1 13020
    8 CONTINUE                                                          E1 13030
C     SAVE ORIGINAL NETWORK                                             E1 13040
      DO 10 I = 1,NR                                                    E1 13050
      DO 10 J = 1,NR                                                    E1 13060
      OINC$M(I,J) = INC$MX(I,J)                                         E1 13070
   10 CONTINUE                                                          E1 13080
C     SAVE ORIGINAL OUTPUTS                                             E1 13090
C     SAVE ORIGINAL OUTPUTS IN (2,1280) FORMAT                          E1 13100
      START = (N*N2) + 1                                                E1 13110
      STOP = (NM*N2)                                                    E1 13120
      DO 13 I = START, STOP                                             E1 13130
      P$(2,I) = P$(1,I)                                                 E1 13140
   13 CONTINUE                                                          E1 13150
C     SAVE ORIGINAL OUTPUTS IN CODED (40,32) FORMAT                     E1 13160
      DO 27 I = N1,NM                                                   E1 13170
      X = (  I-1) * N2                                                  E1 13180
      DO 28 J = 1,N2                                                    E1 13190
      Y = P$(1,X+J)                                                     E1 13200
      IF(Y)30,31,32                                                     E1 13210
C     COMPONENT IS DON'T CARE  (I.E., -1)                               E1 13220
   30 ORGOUT(I,J) = 0                                                   E1 13230
      GOTO 28                                                           E1 13240
C     COMPONENT IS LOGICAL ZERO                                         E1 13250
   31 ORGOUT(I,J) = -100                                                E1 13260
      GO TO 28                                                          E1 13270
C     COMPONENT IS LOGICAL ONE                                          E1 13280
   32 ORGOUT(I,J) = 1                                                   E1 13290
   28 CONTINUE                                                          E1 13300
   27 CONTINUE                                                          E1 13310
C                                                                       E1 13320
C     BLOCK  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C E1 13330
C                                                                       E1 13340
      PCOUNT = 0                                                        E1 13350
   11 PCOUNT = PCOUNT + 1                                               E1 13360
      IF(PCOUNT.GT. R)GO TO 23                                          E1 13370
      PCO = PORDER(PCOUNT)                                              E1 13380
      IF(ONES(PCO).EQ.0)GO TO 11                                        E1 13390
      IF(PCO.LE.NM)GO TO 11                                             E1 13400
C     ERRORS UNCORRECTABLE, RESTORE NETWORK, TRY AGAIN                  E1 13410
      DO 19 I = 1,NR                                                    E1 13420
      DO 19 J = 1,NR                                                    E1 13430
      INC$MX(I,J) = OINC$M(I,J)                                         E1 13440
   19 CONTINUE                                                          E1 13450
C     REMOVE GATE PCO FROM THE NETWORK                                  E1 13460
      DO 12 I = 1,NR                                                    E1 13470
      IF(INC$MX(I,PCO).EQ.0)GO TO 34                                    E1 13480
      INC$MX(I,PCO) = 0                                                 E1 13490
   34 IF(INC$MX(PCO,I).EQ.0) GO TO 12                                   E1 13500
      INC$MX(PCO,I) = 0                                                 E1 13510
   12 CONTINUE                                                          E1 13520
C     UPDATE GATE OUTPUTS FOR ALTERED NETWORK                           E1 13530
C                                                                       E1 13540
C     BLOCK  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D E1 13550
C                                                                       E1 13560
   33 CALL SUBNET                                                       E1 13570
      CALL PVALUE                                                       E1 13580
      CALL UNNECE                                                       E1 13590
C     RESTORE GSMALL FOR OUTPUT GATES                                   E1 13600
      DO 29 I = N1,NM                                                   E1 13610
      DO 29 J = 1, N2                                                   E1 13620
      GSMALL(I,J) = ORGOUT(I,J)
   29 CONTINUE
      ERRORS = 0
      DO 24 I=1,N2
   24 EP(I) = 0
      DO 14 I = 1,M
      NI = N + I
      X = (NI - 1) * N2
      DO 15 J = 1,N2
      IF(GSMALL(NI,J))16,15,17
C     CASE WHERE REQUIREMENT IS A ZERO
   16 IF(P$(1,X+J).EQ.0)GO TO 15
C     CASE OF ONE WITH ERROR
      GSMALL(NI,J) = 1001
      ERRORS = ERRORS + 1
      EP(J) = 1
      GO TO 15
C     CASE WHERE REQUIREMENT IS A ONE
   17 IF(P$(1,X+J).EQ.1)GO TO 15
C     CASE OF ZERO WITH ERROR
      GSMALL(NI,J) = -1100
      ERRORS = ERRORS + 1
      EP(J) = 1
   15 CONTINUE
   14 CONTINUE
      IF(ERRORS.EQ.0)WORKED = 1
      IF(ERRORS.EQ.0) RETURN
      NEP = 0
      DO 25 I = 1,N2
      IF(EP(I).EQ.0) GO TO 25
      NEP = NEP + 1
   25 CONTINUE
      IF(NEP.GT.NEPMAX) GO TO 11
C
C     BLOCK  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E
C
      CALL POT
C     'POT' IS A SUBROUTINE THAT GENERATES THE POTENTIAL OUTPUT TABLE
C
C     BLOCK  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F
C
C     SAVE NEW NETWORK OUTPUTS
      DO 18 J = 1,N2
      DO 18 I = N1,NM
      NETOUT(I,J) = GSMALL(I,J)
   18 CONTINUE
      CALL FORMG3
      CALL INITGS
      CALL RCEC(&11,&33)
C
C     BLOCK  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I
C
C     CASE OF ALL POSSIBLE GATE REMOVALS EXHAUSTED
   23 DO 26 I = 1,NR
      DO 26 J = 1,NR
      INC$MX(I,J) = OINC$M(I,J)
   26 CONTINUE
      CALL SUBNET
      CALL PVALUE
      RETURN
      END
      SUBROUTINE RCEC(*,*)                                              E1 14240
C                                                                       E1 14250
C     SUBROUTINE FOR REMOVING CONNECTIONS BY ERROR COMPENSATION         E1 14260
C                    -          -               -     -                 E1 14270
C                                                                       E1 14280
C                                                                       E1 14290
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.   E1 14300
C                                                                       E1 14310
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                            E1 14320
      COMMON NERMAX                                                     E1 14330
      COMMON    N               , M               , A               , B E1 14340
     1    ,     R               , N2              , N1              , NR E1 14350
     2    ,     NM              , KFLAG           , JFLAG           , COST E1 14360
     3    ,     LEVM            , NRN2            , NM1             , NN2 E1 14370
      COMMON    ISUCC(40,40)    , LISUCC(40)      , IPRED(40,40)    , LIPRED(40) E1 14380
     1    ,     INC$MX(40,40)   , SUC$MX(40,40)   , P$(2,1280)      , UNAME(40) E1 14390
     2    ,     GLEVEL(40)      , LGLIST(40)      , HLIST(40,40)    , TIME E1 14400
      COMMON    T               , RTCONN(100)     , S               , RSCONN(100) E1 14410
      COMMON    IFLAG           ,POINTA           ,ESS1S(40)        ,F$1(32) E1 14420
     1    ,F$UB1                ,INPTCV(32)       ,LISTC(40)        ,POINTC E1 14430
     2    ,LISTL(40)            ,POINTL           ,ORIGIN(40)       ,IPATH(40) E1 14440
     3    ,PO$NTR               ,VF$1(32)         ,VF$UB1           ,GSMALL(40,32)E1 14450
      COMMON    POTAB(200,42),PPOTAB(40)          ,LPOTAB(40)       ,NRPLC(2) E1 14460
     1    ,RPLC(2,40)           ,IDX0(32)         ,IDX0E(32)        ,IDX1(32) E1 14470
     2    ,IDX1E(32)            ,SUMP(32)         ,SETT1(32)        ,NOT1 E1 14480
     3    ,SETS1(40)            ,NOS1             ,SETS(40)         ,NOS E1 14490
     4    ,STS                  ,SUMS2(32)        ,SETS2(200)       ,NOS2 E1 14500
     5    ,LIP                  ,NODE             ,KEYA             ,KEYB E1 14510
     6    ,NO0                  ,NO1              ,NO1E             ,$GT E1 14520
     7    ,$LTH                 ,$PW              ,$NDE             ,GI E1 14530
      COMMON                    NOT1SV          ,NOS1SV          ,LMTS2 E1 14540
      DIMENSION  LISCND(200),                 DI(10), DIO(10), BI(200), E1 14550
     1 BIO(200),IDXK(32)                                                E1 14560
      DIMENSION ORDRP4(40)                                              E1 14570
C     SELECT ONE GATE, GI, WHOSE CSPFE IS ALREADY CALCULATED.           E1 14580
C     ACCORDING TO THE ORDER P1                                         E1 14590
      GI = 0                                                            E1 14600
      LEV = 0                                                           E1 14610
  100 LEV = LEV + 1                                                     E1 14620
      IF(LEV.GE.LEVM) RETURN1                                           E1 14630
      LGL = LGLIST(LEV)                                                 E1 14640
      LG = 0                                                            E1 14650
  120 LG = LG + 1                                                       E1 14660
      IF(LG.GT.LGL)GO TO 100                                            E1 14670
      GI = HLIST(LG,LEV)                                                E1 14680
      GO TO 150                                                         E1 14690
C     IN CASE THE ORDERING P1 HAS BEEN CHANGED DURING PREVIOUS CALCULA- E1 14700
C     TION FOR GI (THE LEVEL OF GI CAN NEVER CHANGE), FIND THE NEXT     E1 14710
C     GATE OF GI ACCORDING TO THE CURRENT ORDERING P1                   E1 14720
  130 LGL = LGLIST(LEV)                                                 E1 14730
      LG = 0                                                            E1 14740
  140 LG = LG + 1                                                       E1 14750
C     IF(LG.GT.LGL)GO TO 100      CAN NEVER HAPPEN                      E1 14760
      IF(GI.EQ.HLIST(LG,LEV))GO TO 120                                  E1 14770
      GO TO 140                                                         E1 14780
C**** CONSIDER INPUTS OF GATE GI                                         E1 14790
  150 IF(GI.LE.N) GO TO 120                                             E1 14800
      IF(GI.GT.NM.AND.GLEVEL(GI).LE.1)GO TO 120                         E1 14810
      LIP =LIPRED(GI)                                                   E1 14820
```

```
C**** INITIALIZE ARRAY (VECTOR) SUMP AND SUMS2
      DO 160 TH=1,N2
        SUMS2(TH) = 0
  160   SUMP(TH) = 0
C**** SUM-UP ALL INPUTS OF GI
      DO 180 LP =1,LIP
        GJ = IPRED(LP,GI)
        BSGJ = (GJ-1)*N2
        DO 170 TH=1,N2
  170     SUMP(TH) = SUMP(TH) + P$(1,BSGJ + TH)
  180   CONTINUE
C**** LIST 0,0-ERROR,1,1-ERROR COMPONENTS OF CSPFE OF GI
      NOO = 0
      NO1 = 0
      NOOE = 0
      NO1E = 0
      DO 230 TH=1,N2
        IF(GSMALL(GI,TH))190,230,210
  190   IF(GSMALL(GI,TH).LE.-1000)GO TO 200
        NOO = NOO + 1
        IDXO(NOO) = TH
        GO TO 230
  200   NOOE = NOOE + 1
        IDXOE(NOOE) = TH
        GO TO 230
  210   IF(GSMALL(GI,TH).GE.1000)GO TO 220
        NO1 = NO1 + 1
        IDX1(NO1) = TH
        GO TO 230
  220 NO1E = NO1E + 1
        IDX1E(NO1E) = TH
  230   CONTINUE
C**** REMOVE REDUNDANT CONNECTIONS BY CALLING FORC ****
C     LIP = LIPRED(GI)
      KEYA = 0
      KEYB = 0
      DO 350 LP=1,LIP
        GJ = IPRED(LP,GI)
        CALL FORC(GJ)
  350   CONTINUE
C**** COMPARE SUMP WITH THE OUTPUT OF GI TO CHECK WHETHER ALL ERRORS
C     ARE CORRECTED OR NOT ****
      IF(NOOE.EQ.0) GO TO 380
      NOE=0
      DO 370 IDX=1,NOOE
        IF(SUMP(IDXOE(IDX)).GE.1) GO TO 360
        NO1=NO1+1
        IDX1(NO1)=IDXOE(IDX)
        GO TO 370
  360   NOE=NOE+1
        IDXOE(NOE)=IDXOE(IDX)
  370   CONTINUE
      NOOE=NOE
C**** ALL ERRORS IN GI ARE CORRECTED ****
  380 IF(NOOE.EQ.0.AND.NO1E.EQ.0) GO TO 1400
C**** CALCULATION OF EFFECTIVELY CONNECTABLE FUNCTIONS FOR GI
C**** ACCESS EACH GATE IN THE NETWORK
  400 CNODT = 0
      DO 490 GR =1,NR
        IF(GLEVEL(GR).EQ.1.AND.GR.GT.NM)GO TO 490
        IF(INC$MX(GR,GI).GE.1.OR.GI.EQ.GR.OR.SUC$MX(GI,GR).GE.1)GOTO490
```

```
C**** IF GR IS CONNECTED TO ALL SUCCESSORS OF GI, IT IS NOT A CANDIDATE E1 15440
        LIS = LISUCC(GI)                                                E1 15450
        IF(LIS.EQ.0) GO TO 415                                          E1 15460
        DO 410 LI=1,LIS                                                 E1 15470
          IF(INC$MX(GR,ISUCC(LI,GI)).LE.0) GO TO 415                    E1 15480
  410     CONTINUE                                                      E1 15490
        GO TO 490                                                       E1 15500
  415   CONTINUE                                                        E1 15510
C****   CHECK THE POSSIBLE OUTPUTS OF THAT GATE ****                    E1 15520
        PA = PPOTAB(GR)                                                 E1 15530
        IF(PA.LE.0) GOTO 490                                            E1 15540
        PB = LPOTAB(GR)                                                 E1 15550
        DO 485 PTR=PA,PB                                                E1 15560
          LTH = POTAB(PTR,$LTH)                                         E1 15570
          IF(LTH.LE.0) GO TO 425                                        E1 15580
          DO 420 LT=1,LTH                                               E1 15590
            IF(SUC$MX(GI,POTAB(PTR,$LTH+LT)).GE.1)GO TO 485             E1 15600
            IF(INC$MX(POTAB(PTR,$LTH+LT),GI).GE.1) GO TO 485            E1 15610
  420       CONTINUE                                                    E1 15620
C****     CHECK THE EFFECTIVELY CONNECTABILITY ****                     E1 15630
  425     IF(NO1.EQ.0) GO TO 435                                        E1 15640
          DO 430 IDX=1,NO1                                              E1 15650
            IF(POTAB(PTR,IDX1(IDX)).NE.0) GO TO 485                     E1 15660
  430       CONTINUE                                                    E1 15670
C****     POTAB(PTR) IS EFFECTIVELY CONNECTABLE TO GI ****              E1 15680
C****     CALCULATE THE PREFERENCE WEIGHT ****                          E1 15690
  435     PW = 0                                                        E1 15700
          IF(NO0.EQ.0) GO TO 445                                        E1 15710
          DO 440 IDX=1,NO0                                              E1 15720
            IF(POTAB(PTR,IDX0(IDX)).GE.1)PW = PW + 1                    E1 15730
  440       CONTINUE                                                    E1 15740
  445     IF(NO1E.EQ.0) GO TO 460                                       E1 15750
          DO 450 IDX=1,NO1E                                             E1 15760
            IF(POTAB(PTR,IDX1E(IDX)).GE.1)PW = PW + 1                   E1 15770
  450       CONTINUE                                                    E1 15780
C****     SORT CANDIDATES IN ORDER ACCORDING TO PW                      E1 15790
  460     IF(CNDDT.EQ.0)GO TO 475                                       E1 15800
          DO 470 CR=1,CNDDT                                             E1 15810
            CND = CNDDT-CR+1                                            E1 15820
            IF(PW.LE.POTAB(LISCND(CND),$PW))GO TO 480                   E1 15830
            LISCND(CND+1) = LISCND(CND)                                 E1 15840
  470       CONTINUE                                                    E1 15850
  475   CND = 0                                                         E1 15860
  480   LISCND(CND+1) = PTR                                             E1 15870
        POTAB(PTR,$PW) = PW                                             E1 15880
        CNDDT = CNDDT + 1                                               E1 15890
  485   CONTINUE                                                        E1 15900
  490   CONTINUE                                                        E1 15910
      IF(CNDDT.EQ.0) GO TO 1490                                         E1 15920
C**** CLASSIFY CANDIDATES INTO BI,DI,BIO,AND DIO *****                   E1 15930
      NODI = 0                                                          E1 15940
      NOBI = 0                                                          E1 15950
      NODIO = 0                                                         E1 15960
      NOBIO = 0                                                         E1 15970
      DO 650 NC =1, CNDDT                                               E1 15980
        NOONEE = 0                                                      E1 15990
        PTR = LISCND(NC)                                                E1 16000
        IF(NOOE.EQ.0)GO TO 630                                          E1 16010
        DO 610 NO=1,NOOE                                                E1 16020
          IF(POTAB(PTR,IDXOE(NO)).LE.0) GO TO 610                       E1 16030
          NOONEE = NOONEE + 1                                           E1 16040
  610     CONTINUE
        IF(NOONEE.EQ.0)GO TO 630
        POTAB(PTR,$NOE) = NOONEE
        IF(POTAB(PTR,$GT).GT.N)GO TO 620
        NODI = NODI + 1
        DI(NODI) = PTR
        GO TO 650
C****   PUT PTR INTO TABLE BI AND SORT IT ACCORDING TO ORDER Q1(NOONEE)
  620   CONTINUE
        IF(NOBI.EQ.0) GO TO 624
        DO 623 NO=1,NOBI
          NOB = NOBI - NO + 1
          IF(POTAB(  BI(NOB),$NOE).LE.NOONEE) GO TO 627
          BI(NOB+ 1) = BI(NOB)
  623     CONTINUE
  624   NOB = 0
  627   BI(NOB+1) = PTR
        NOBI = NOBI + 1
        GO TO 650
  630   POTAB(PTR,$NOE) = 0
        IF(POTAB(PTR,$GT).GT.N)GO TO 640
        NODIO = NODIO + 1
        DIO(NODIO) = PTR
        GO TO 650
  640   NOBIO = NOBIO + 1
        BIO(NOBIO) = PTR
  650   CONTINUE
C**** CALCULATE SET S2 *****
      IF(NOBIO.EQ.0)GO TO 770
      DO 760 NO=1,NOBIO
  760   SETS2(NO) = BIO(NO)
  770 IF(NODIO.EQ.0)GO TO 790
      DO 780 NO=1,NODIO
  780   SETS2(NOBIO+NO) = DIO(NO)
  790 IF(NOBI.EQ.0)GO TO 810
      DO 800 NO=1,NOBI
  800   SETS2(NOBIO+NODIO+NO) = BI(NO)
C**** CALL PROCEDURES TO REPLACE EXTERNAL VARIABLES *****
  810 NOS2 = NOBIO + NODIO + NOBI
      LMTS2=NOS2
      IF(NOS2.EQ.0) GO TO 1400
      IF(NOOE.EQ.0) GO TO 1210
C**** CALCULATE SUMS2 *****
      DO 832 NO=1,NOS2
        PTR = SETS2(NO)
        DO 830 TH=1,N2
  830     SUMS2(TH) = SUMS2(TH) + POTAB(PTR,TH)
  832   CONTINUE
C**** REPLACEMENT OF EXTERNAL VARIABLES *****
C**** CALCULATE SET S *****
      NOS = 0
      DO 750 LP = 1,LIP
        GP = IPRED(LP,GI)
        IF(INC$MX(GP,GI).LE.0.OR.GP.GT.N)GO TO 750
        BSGP = (GP-1)*N2
        DO 700 NO=1,NOOE
          IF(P$(1,BSGP+IDXOE(NO)).GE.1) GO TO 710
  700     CONTINUE
        GO TO 750
  710   NOS = NOS + 1
        SETS(NOS) = GP
  750   CONTINUE
      STS = 1
      IF(NDS.LE.0)GO TO 835
      NDT1SV=0
      NDS1SV=0
      CALL CALS1
      CALL RPLCF
C**** REPLACEMENT OF INTERNAL FUNCTIONS WITH 1 ERROR *****
  835 BRPLC = 0
      NDS1=0
      NDT1=0
      NOS = 0
      CALL ORDRQ2
      P2 = NRPLC(2)
      IF(P2.EQ.0) GO TO 995
      DO 990 PR=1,P2
        GP = RPLC(2,PR)
        NDS = NDS + 1
        SETS(NDS) = GP
        BSGP = (GP - 1)*N2
C****   CALCULATE THE ERROR POSITIONS OF WEIGHT 1 IN GP *****
        NOK = 0
        DO 840 NO=1,NOOE
          TH = IDXOE(NO)
          IF(SUMP(TH).NE.1.OR.P$(1,BSGP+TH).NE.1)GO TO 840
          NOK = NOK + 1
          IDXK(NOK) = TH
  840     CONTINUE
C****   CALCULATE SETS4  *****
        DO 940 NK=1,NOK
          TH = IDXK(NK)
C****     IF SETS2(*)=1000+PTR, IT IS NOT IN SET S4    *****
          DO 870 SR=1,NOS2
            IF(SETS2(SR).GT.1000) GO TO 870
            IF(POTAB(SETS2(SR),TH).EQ.0) GO TO 861
C****       UPDATE SUMS2 *****
            DO 860 TT=1,N2
  860         SUMS2(TT) = SUMS2(TT) - POTAB(SETS2(SR),TT)
            SETS2(SR) = SETS2(SR) + 1000
              GO TO 870
C**** TEMPORARILY PUT SETS2(SR) INTO SETS4. PROHIBIT FUNCTIONS WHICH
C     HAS SAME OUTPUT GATE AS SETS2(SR)
  861       CONTINUE
            IF(BRPLC.EQ.1) GO TO 870
            GT=POTAB(SETS2(SR),$GT)
            DO 865 SRR=1,NOS2
              IF(SRR.EQ.SR) GO TO 865
              IF(SETS2(SRR).GT.1000) GO TO 865
              IF(POTAB(SETS2(SRR),$GT).EQ.GT) GO TO 8617
              IF(POTAB(SETS2(SRR),$LTH).EQ.0) GO TO 865
              MRUN=POTAB(SETS2(SRR),$LTH)
              DO 8615 RUN=1,MRUN
                IF(POTAB(SETS2(SRR),$LTH+RUN).EQ.GT) GO TO 8617
 8615         CONTINUE
              GO TO 865
 8617         CONTINUE
C**** UPDATE SUMS2(*)
              DO 863 TT=1,N2
  863           SUMS2(TT)=SUMS2(TT)-POTAB(SETS2(SRR),TT)
              SETS2(SRR)=SETS2(SRR)+1000
  865       CONTINUE
  870       CONTINUE
C****     CHECK WHETHER OR NOT ELEMENTS IN SETS4 COVER ALL ESSENTIAL
C         ONES OF GP *****
          DO 900 NO=1,NOO
            TT=IDXO(NO)
            IF(P$(1,BSGP+TT).NE.1.OR.SUMP(TT).NE.1) GO TO 900
            IF(SUMS2(    TT    ).EQ.0)GO TO 930
  900       CONTINUE
C****     S4 COVERS ALL ESSENTIAL ONES *****
C         UPDATE SETS2,SETS1, AND T1
          BRPLC = 1
          DO 910 SR=1,NOS2
            IF(SETS2(SR).GT.1000.AND.SETS2(SR).LT.2000)
     1      SETS2(SR) = SETS2(SR) + 1000
  910       CONTINUE
          NOS1 = NOS1 + 1
          SETS1(NOS1) = GP
          SETS(NOS) = 2000 + GP
          DO 915 NO=1,NOO
            TT=IDXO(NO)
            IF(P$(1,BSGP+TT).NE.1.OR.SUMP(TT).NE.1) GO TO 915
            NOT1 = NOT1 + 1
            SETT1(NOT1) = TT
  915       CONTINUE
C****     UPDATE SUMP  *****
          DO 920 TH=1,N2
            SUMP(TH)=SUMP(TH) - P$(1,BSGP+TH)
  920       CONTINUE
          GO TO 990
C****     RESET SETS4 ****
  930     DO 935 NO=1,NOS2
            IF(SETS2(NO).LT.1000.OR.SETS2(NO).GT.2000)GO TO 935
            SETS2(NO) = SETS2(NO) - 1000
C****       UPDATE SETS2 (THIS ELEMENT BECOMES ACTIVE AGAIN)
            DO 932 TH=1,N2
  932         SUMS2(TH) = SUMS2(TH) + POTAB(SETS2(NO),TH)
  935       CONTINUE
  940     CONTINUE
  990   CONTINUE
C**** PUT RPLC(1,*) INTO SET S *****
  995 STS1 = NOS + 1
      P2 = NRPLC(1)
      IF(P2.EQ.0)GO TO 1005
      DO 1000 PR=1,P2
        NOS = NOS + 1
        SETS(NOS) = RPLC(1,PR)
 1000   CONTINUE
 1005 IF(BRPLC.EQ.0)GO TO 1010
      NOT1SV=NOT1
      NOS1SV=NOS1
      CALL CALS1
      CALL RPLCF
C**** REPLACEMENT BY BIO AND DIO *****
 1010 NOS2 = NOBIO + NODIO
C**** UPDATE SUMS2(*) TO CONTAIN THE CURRENT ELEMENTS IN SETS2 ONLY ****
      IF(NOBI.EQ.0) GO TO 1060
      S1 = NOBIO + NODIO + 1
      S2 = S1 + NOBI - 1
      DO 1050 NO=S1,S2
        PTR = SETS2(NO)
        IF(PTR.GT.1000) GO TO 1050
      DO 1040 TH=1,N2
 1040     SUMS2(TH) = SUMS2(TH) - POTAB(PTR,TH)
 1050   CONTINUE
 1060 CONTINUE
      IF(NOS2 .EQ.0) GO TO 1100
      STS = STS1
      IF(STS.GT.NOS)GO TO 1100
      NOS1SV=0
      NOT1SV=0
      CALL CALS1
      CALL SPLCF
C**** COMPENSATION OF 1 ERRORS OF CSPFE OF GI
 1100 CONTINUE
C**** SUM-UP ADDED FUNCTIONS *****
C     MODIFY SUMS2 TO CONTAIN ONLY ADDED CONNECTIONS
      IF(NOS2.EQ.0) GO TO 1175
      DO 1170 NO=1,NOS2
        PTR = SETS2(NO)
        IF(PTR.GT.1000)GO TO 1170
        DO 1150 TH=1,N2
          SUMS2(TH) = SUMS2(TH) - POTAB(PTR,TH)
 1150   CONTINUE
 1170 CONTINUE
 1175 CONTINUE
      NOS2 = NOBIO + NODIO + NOBI
C**** LIST UNCOVERED 1-ERROR COMPONENTS *****
      IF(NO1E.EQ.0)GO TO 1400
      NOTO = 0
      DO 1200 NO=1,NO1E
        TH = IDX1E(NO)
        IF(SUMS2(TH).GE.1)GO TO 1180
        NOTO = NOTO +1
        GO TO 1200
C****   TH IS COVERED ALREADY (IDX1E(*)>1000 : COVERED)
 1180   IDX1E(NO) = 1000 + TH
 1200 CONTINUE
      IF(NOTO.EQ.0)GO TO 1400
      GO TO 1220
 1210 NOTO = NO1E
C**** RESTRICTIONS ON S2 AS STATED IN ALGORITHM MAY BE INSERTED HERE
C**** CHECK ACTIVE FUNCTIONS IN S2
 1220 CONTINUE
      DO 1300 NO=1,NOS2
        PTR=SETS2(NO)
        IF(PTR.GT.2000)GO TO 1300
C****   CHECK WHETHER THIS FUNCTION CAN COMPENSATE SOME ONE-ERRORS
C       OR NOT
        NOTOO = NOTO
        DO 1230 NOE=1,NO1E
          IF(IDX1E(NOE).GT.1000)GO TO 1230
          IF(POTAB(PTR,IDX1E(NOE)).NE.1)GO TO 1230
          NOTO = NOTO - 1
          IDX1E(NOE) = 1000 + IDX1E(NOE)
 1230   CONTINUE
        IF(NOTO.EQ.NOTOO)GO TO 1300
        DO 1235 TH=1,N2
 1235     SUMP(TH)=SUMP(TH)+POTAB(SETS2(NO),TH)
        SETS2(NO) = 5000 + PTR
        CALL CONECT(PTR)
C**** PROHIBIT FUNCTIONS WHICH HAS SAME OUTPUT GATE AS PTR ****
        IF(NO.EQ.NOS2) GO TO 1300
        SR1=NO+1                                                         E1 18490
        GT=POTAB(PTR,$GT)                                                E1 18500
        DO 1250 SR=SR1,NOS2                                              E1 18510
          IF(SETS2(SR).GT.2000) GO TO 1250                               E1 18520
          IF(POTAB(SETS2(SR),$GT).EQ.GT) GO TO 1245                      E1 18530
          IF(POTAB(SETS2(SR),$LTH).EQ.0) GO TO 1250                      E1 18540
          MRUN=POTAB(SETS2(SR),$LTH)                                     E1 18550
          DO 1240 RUN=1,MRUN                                             E1 18560
            IF(POTAB(SETS2(SR),$LTH+RUN).EQ.GT) GO TO 1245               E1 18570
 1240     CONTINUE                                                       E1 18580
          GO TO 1250                                                     E1 18590
 1245     SETS2(SR)=SETS2(SR)+2000                                       E1 18600
 1250   CONTINUE                                                         E1 18610
 1300 CONTINUE                                                           E1 18620
 1400 CONTINUE                                                           E1 18630
C**** ADDING EXTERNAL VARIABLES TO GI *****                              E1 18640
      DO 1480 GP=1,N                                                     E1 18650
        IF(INC$MX(GP,GI).GE.1) GO TO 1480                                E1 18660
        BSGP=(GP-1)*N2                                                   E1 18670
C**** CHECK CONNECTABILITY *****                                         E1 18680
        DO 1410 IDX=1,NO1                                                E1 18690
          IF(P$(1,BSGP+IDX1(IDX)).GE.1) GO TO 1480                       E1 18700
 1410   CONTINUE                                                         E1 18710
        IF(NO0E.EQ.0) GO TO 1430                                         E1 18720
        DO 1420 IDX=1,NO0E                                               E1 18730
          TH=IDX0E(IDX)                                                  E1 18740
          IF(P$(1,BSGP+TH).GE.1.AND.SUMP(TH)           .LE.1) GO TO 1480 E1 18750
 1420 CONTINUE                                                           E1 18760
C**** CHECK WHETHER OR NOT IT COVERS 0 OR 1-ERROR COMPONENTS ****        E1 18770
 1430   CONTINUE                                                         E1 18780
        DO 1440 IDX=1,NO0                                                E1 18790
          IF(P$(1,BSGP+IDX0(IDX)).GE.1) GO TO 1460                       E1 18800
 1440   CONTINUE                                                         E1 18810
        IF(NO1E.EQ.0) GO TO 1480                                         E1 18820
        DO 1450 IDX=1,NO1E                                               E1 18830
          TH=IDX1E(IDX)                                                  E1 18840
          IF(TH.GT.1000) TH=TH-1000                                      E1 18850
          IF(P$(1,BSGP+TH).GE.1) GO TO 1460                              E1 18860
 1450   CONTINUE                                                         E1 18870
        GO TO 1480                                                       E1 18880
C**** GP CAN BE CONNECTED TO GI *****                                    E1 18890
 1460   S=S+1                                                            E1 18900
        RSCONN(S)=100*GP+GI                                              E1 18910
        INC$MX(GP,GI)=1                                                  E1 18920
        KEYB=1                                                           E1 18930
 1480 CONTINUE                                                           E1 18940
 1490 IF(KEYA.EQ.0) GO TO 1500                                           E1 18950
C**** SOME ERRORS WERE COMPENSATED IN GI                                 E1 18960
      RETURN2                                                            E1 18970
C**** CALCULATION OF CSPFE FOR INPUTS                                    E1 18980
 1500 CONTINUE                                                           E1 18990
      IF(KEYB.EQ.1)CALL SUBNET                                           E1 19000
      IF(KEYB.EQ.1) CALL UNNECE                                          E1 19010
C**** PROPAGATE ONE, ONE-ERROR, AND ZERO-ERROR COMPONENTS *****          E1 19020
      LIP = LIPRED(GI)                                                   E1 19030
      DO 1600 LP=1,LIP                                                   E1 19040
        GP = IPRED(LP,GI)                                                E1 19050
        IF(GP.LE.N)GO TO 1600                                            E1 19060
        BSGP = (GP-1)*N2                                                 E1 19070
C****   FOR ONE COMPONENTS *****                                         E1 19080
        IF(NO1.EQ.0) GO TO 1530                                          E1 19090
      DO 1520 NO=1,NO1                                                  E1 19100
        IF(GSMALL(GP,IDX1(NO)).EQ.0)GSMALL(GP,IDX1(NO)) = -100          E1 19110
        IF(GSMALL(GP,IDX1(NO)).LT.-1000.AND.GP.GT.NM)                   E1 19120
     1                                  GSMALL(GP,IDX1(NO))= -100       E1 19130
 1520   CONTINUE                                                        E1 19140
C****   FOR ONE-ERROR COMPONENTS *****                                  E1 19150
 1530   IF(NO1E.EQ.0) GO TO 1550                                        E1 19160
        DO 1540 NO=1,NO1E                                               E1 19170
          IF(GSMALL(GP,IDX1E(NO)).EQ.0)GSMALL(GP,IDX1E(NO)) = -1100     E1 19180
 1540   CONTINUE                                                        E1 19190
C****   FOR ZERO-ERROR COMPONENTS *****                                 E1 19200
 1550   IF(NO0E.EQ.0) GO TO 1600                                        E1 19210
        DO 1560 NO=1,NO0E                                               E1 19220
          IF(P$(1,BSGP+IDX0E(NO)).EQ.0)GO TO 1555                       E1 19230
        IF(GSMALL(GP,IDX0E(NO)).EQ.0) GSMALL(GP,IDX0E(NO)) = 1001       E1 19240
          GO TO 1560                                                    E1 19250
 1555     IF(GSMALL(GP,IDX0E(NO)).EQ.0.OR.GSMALL(GP,IDX0E(NO)).LT.-1000)E1 19260
     1        GSMALL(GP,IDX0E(NO))=-100                                 E1 19270
 1560   CONTINUE                                                        E1 19280
 1600 CONTINUE                                                          E1 19290
C**** PROPAGATE ZERO COMPONENTS *****                                    E1 19300
      IF(NO0.EQ.0) GO TO 1800                                           E1 19310
C***********************************************************************E1 19320
C     CALCULATION OF ORDERING P4                                        E1 19330
C     SINCE NO CONNECTIONS ARE ADDED DURING CALCULATION FOR GATE GI,    E1 19340
C     SUMP(*) HAS THE INPUT SUM OF GATE GI                              E1 19350
C**** CALCULATE NUMBER OF 1-ERRORS AND NUMBER OF ESSENTIAL 1-ERRORS IN   E1 19360
C     EACH INPUT OF GI                                                  E1 19370
      DO 1610 LP=1,LIP                                                  E1 19380
        GP=IPRED(LP,GI)                                                 E1 19390
        IF(GP.LE.N) GO TO 1610                                          E1 19400
        NOONEE=0                                                        E1 19410
        NO1EES=0                                                        E1 19420
        IF(NO0E.EQ.0) GO TO 1606                                        E1 19430
        BSGP=(GP-1)*N2                                                  E1 19440
        DO 1605 NO=1,NO0E                                               E1 19450
          TH=IDX0E(NO)                                                  E1 19460
          IF(P$(1,BSGP+TH).GE.1) NOONEE=NOONEE+1                        E1 19470
          IF(SUMP(TH).EQ.1.AND.P$(1,BSGP+TH).EQ.1)NO1EES=NO1EES+1       E1 19480
 1605     CONTINUE                                                      E1 19490
 1606   ORDRP4(GP)=1000000-NO1EES*10000-NOONEE*100+LISJCC(GP)           E1 19500
 1610   CONTINUE                                                        E1 19510
C**** END OF CALCULATION OF ORDERING P4                                  E1 19520
      DO 1700 NO=1,NO0                                                  E1 19530
        TH = IDX0(NO)                                                   E1 19540
        PODRMX = 0                                                      E1 19550
        DO 1620 LP=1,LIP                                                E1 19560
          GP = IPRED(LP,GI)                                             E1 19570
          BSGP = (GP-1)*N2                                              E1 19580
          IF(P$(1,BSGP+TH).EQ.0)GO TO 1620                              E1 19590
C****     IF SOME GATES ARE ALREADY ASSIGNED ONE, DO NOTHING            E1 19600
          IF(GSMALL(GP,TH).EQ.1.OR.GP.LE.N) GO TO 1700                  E1 19610
C****     COMPARE PRIORITY OF CURRENT GATE WITH PREVIOUS HIGHEST ONE    E1 19620
          IF(GP.LE.NM.AND.GSMALL(GP,TH).GT.1000)GO TO 1620              E1 19630
          PODR=ORDRP4(GP)                                               E1 19640
          IF(PODR.LE.PODRMX)GO TO 1620                                  E1 19650
          PODRMX = PODR                                                 E1 19660
          PODRGT = GP                                                   E1 19670
 1620   CONTINUE                                                        E1 19680
        IF(PODRMX.EQ.0)GO TO 1630                                       E1 19690
        GSMALL(PODRGT,TH) = 1                                           E1 19700
        GO TO 1700                                                      E1 19710
C****   THIS ZERO IS COVERED ONLY BY OUTPUT GATES WHICH ARE ALREADY     E1 19720
C       ASSIGNED ONE-ERROR. PROPAGATE ZERO ERROR TO PREDECESSORS WHICH  E1 19730
C       HAVE ZERO COMPONENT                                             E1 19740
 1630   DO 1650 LP=1,LIP                                                E1 19750
          GP = IPRED(LP,GI)                                             E1 19760
          BSGP = (GP-1)*N2                                              E1 19770
          IF(P$(1,BSGP+TH).EQ.1.OR.GP.LE.N)GO TO 1650                   E1 19780
          IF(GSMALL(GP,TH).EQ.0)GSMALL(GP,TH) = -1100                   E1 19790
 1650   CONTINUE                                                        E1 19800
 1700 CONTINUE                                                          E1 19810
 1800 CONTINUE                                                          E1 19820
      GO TO 130                                                         E1 19830
      END                                                               E1 19840


      SUBROUTINE RPLCF                                                  E1 19850
C**** CALCULATE A SUBSET OF SET S2 WHICH CAN REPLACE SET S1              E1 19860
C     SET S2 IS LISTED ACCORDING TO ORDER Q1                        ****E1 19870
C                                                                       E1 19880
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.   E1 19890
C                                                                       E1 19900
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                            E1 19910
      COMMON NERMAX                                                     E1 19920
      COMMON   N            , M            , A            , B           E1 19930
     1    ,    R            , N2           , N1           , NR          E1 19940
     2    ,    NM           , KFLAG        , JFLAG        , COST        E1 19950
     3    ,    LEVM         , NRN2         , NM1          , NN2         E1 19960
      COMMON   ISUCC(40,40) , LISUCC(40)   , IPRED(40,40) , LIPRED(40)  E1 19970
     1    ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)   , UNAME(40)   E1 19980
     2    ,    GLEVEL(40)   , LGLIST(40)   , HLIST(40,40) , TIME        E1 19990
      COMMON   T            , RTCONN(100)  , S            , RSCONN(100) E1 20000
      COMMON   IFLAG        ,POINTA        ,ESS1S(40)     ,F$1(32)      E1 20010
     1    ,F$UB1            ,INPTCV(32)    ,LISTC(40)     ,POINTC       E1 20020
     2    ,LISTL(40)        ,POINTL        ,ORIGIN(40)    ,IPATH(40)    E1 20030
     3    ,POINTR           ,VF$1(32)      ,VF$UB1        ,GSMALL(40,32)E1 20040
      COMMON   POTAB(200,42),PPOTAB(40)    ,LPOTAB(40)    ,NRPLC(2)     E1 20050
     1    ,RPLC(2,40)       ,IDX0(32)      ,IDX0E(32)     ,IDX1(32)     E1 20060
     2    ,IDX1E(32)        ,SUMP(32)      ,SETT1(32)     ,NOT1         E1 20070
     3    ,SETS1(40)        ,NOS1          ,SETS(40)      ,NOS          E1 20080
     4    ,STS              ,SUMS2(32)     ,SETS2(200)    ,NOS2         E1 20090
     5    ,LIP              ,NOOE          ,KEYA          ,KEYB         E1 20100
     6    ,NO0              ,NO1           ,NO1E          , $GT         E1 20110
     7    , $LTH            ,$PW           ,$NOE          ,GI           E1 20120
      COMMON                NOT1SV        ,NOS1SV        ,LMTS2         E1 20130
 6000 IF (NOS1.EQ.0) RETURN                                             E1 20140
      DO 6200 NO=1,NOS2                                                 E1 20150
      IF(SETS2(NO).GT.2000) GO TO 6200                                  E1 20160
      PTR=SETS2(NO)                                                     E1 20170
C**** CHECK ESSENTIAL ONES IN S1                                         E1 20180
      KEYT2=0                                                           E1 20190
      DO 6100 NOT=1,NOT1                                                E1 20200
      IF(SETT1(NOT).GT.1000) GO TO 6100                                 E1 20210
      TH=SETT1(NOT)                                                     E1 20220
      IF(POTAB(PTR,TH).LE.0)GO TO 6100                                  E1 20230
      KEYT2=1                                                           E1 20240
      SETT1(NOT)=1000+TH                                                E1 20250
 6100 CONTINUE                                                          E1 20260
C**** IF T2 IS EMPTY THIS FUNCTION IS NOT IN SET S3                      E1 20270
      IF(KEYT2.LE.0) GO TO 6200                                         E1 20280
C**** PROHIBIT FUNCTIONS IN S2 WHICH USE SAME OUTPUT GATE AS PTR *****   E1 20290
```

```
      IF(NOT1SV.GT.0) GOTO 6180
C                   NOT1SV>0 => BRPLD>0 => SETS2 IS COMPATIBLE
      SW=0
      GT=POTAB(PTR,$GT)
      DO 6150 IDX=1,LMTS2
        IF(IDX.EQ.NO) GO TO 6150
        IF(SETS2(IDX).GT.2000) GO TO 6150
        IF(POTAB(SETS2(IDX),$GT).EQ.GT) GO TO 6140
        IF(POTAB(SETS2(IDX),$LTH).EQ.0) GO TO 6150
        MRUN=POTAB(SETS2(IDX),$LTH)
        DO 6120 RUN=1,MRUN
          IF(POTAB(SETS2(IDX),$LTH+RUN).EQ.GT) GO TO 6140
 6120   CONTINUE
        GO TO 6150
 6140   CONTINUE
        IF(IDX.GT.NOS2) GO TO 6145
        SW=1
C**** UPDATE SUMS2 *****
        DO 6130 TH=1,N2
 6130   SUMS2(TH)=SUMS2(TH)-POTAB(SETS2(IDX),TH)
 6145   SETS2(IDX)=3000+SETS2(IDX)
 6150 CONTINUE
      IF(SW.EQ.0) GO TO 6180
C**** CHECK WHETHER S2 STILL COVER ALL ESSENTIAL ONES OR NOT *****
C     IF YES CONTINUE THIS PROCEDURE, OTHERWISE RECALCULATE S1 AND
C     REPEAT THIS PROCEDURE
      DO 6160 NOT=1,NOT1
        IF(SETT1(NOT).GT.1000) GO TO 6160
        IF(SUMS2(SETT1(NOT)).LE.0) GO TO 6190
 6160 CONTINUE
 6180 SETS2(NO)=4000+PTR
      GO TO 6200
 6190 CONTINUE
      DO 6195 TH=1,N2
 6195   SUMS2(TH)=SUMS2(TH)-POTAB(PTR,TH)
      SETS2(NO)=PTR+2000
      GO TO 6500
 6200 CONTINUE
C**** REPLACE FUNCTIONS IN S1 BY FUNCTIONS IN S2 ****
      DO 6300 NO=1,LMTS2
        PTR=SETS2(NO)
        IF(PTR.GT.5000) GO TO 6300
        IF(PTR.GT.4000) GO TO 6210
        IF(PTR.GT.3000) GO TO 6290
        GO TO 6300
C**** ADDING POTAB(PTR,*) TO SUMP ****
 6210   CONTINUE
        DO 6220 TH=1,N2
 6220     SUMP(TH)=SUMP(TH)+POTAB(PTR-4000,TH)
C**** RECORD THIS FUNCTION HAS BEEN CONNECTED  (SETS2(*)>5000) *****
        SETS2(NO)=1000+PTR
C**** CONNECT THIS FUNCTION TO GI AND MAKE OTHER CONNECTIONS NECESSARY
C     FOR REALIZING THIS FUNCTION                                  *****
        CALL CONECT(PTR-4000)
        GO TO 6300
C**** THIS FUNCTION CAN NO LONGER BE USED TO REPLACE OTHER FUNCTIONS ***
 6290   SETS2(NO)=PTR-1000
 6300 CONTINUE
C**** DISCONNECTION PREDECESSORS OF GI IN SET S1
      DO 6400 NO=1,NOS1
      GP=SETS1(NO)
      INC$MX(GP,GI)=0
      T=T+1
      RTCONN(T)=100*GP+GI
 6400 CONTINUE
C**** MAKE PERMANENT CHANGES IN SET S (S>2000 : REMOVED )
C****                          (2000>S>1000 : TEMPORARILY REMOVED )
      DO 6450 NO=1,NOS
        IF(SETS(NO).LT.2000.AND.SETS(NO).GT.1000) SETS(NO)=SETS(NO)+1000
 6450 CONTINUE
      RETURN
C**** SET S1 IS NOT REPLACABLE BY S2. RECALCULATE SET S1
C     PROHIBIT THE FUNCTIONS WHICH CAUSED THIS SITUATION ****
 6500 CONTINUE
      DO 6550 NO=1,LMTS2
        PTR=SETS2(NO)
        IF(PTR.GT.5000) GO TO 6550
        IF(PTR.GT.4000) GO TO 6510
        IF(PTR.GT.3000) GO TO 6520
        GO TO 6550
C**** MAKE THIS ELEMENT ACTIVE *****
 6510   SETS2(NO)=PTR-4000
        GO TO 6550
 6520    SETS2(NO)=PTR-3000
C**** MAKE THIS ELEMENT ACTIVE ****
        IF(NO.GT.NOS2)GO TO 6550
        DO 6530  TH=1,N2
 6530     SUMS2(TH)=SUMS2(TH)+POTAB(SETS2(NO),TH)
 6550 CONTINUE
C**** ADD BACK ELEMENTS OF SET S1 TO SUMP
      DO 6600 NO=1,NOS1
        GP=SETS1(NO)
        BSGP=(GP-1)*N2
        DO 6580 TH=1,N2
 6580     SUMP(TH)=SUMP(TH)+P$(1,BSGP+TH)
 6600 CONTINUE
C**** RECONSTRUCT SET S *****
      DO 6650 NO=1,NOS
        IF(SETS(NO).GT.2000) GO TO 6650
        IF(SETS(NO).LT.1000) GO TO 6650
        SETS(NO)=SETS(NO)-1000
 6650 CONTINUE
      CALL CALS1
      GO TO 6000
      END


      SUBROUTINE SUBNET
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM
C
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
      COMMON NEPMAX
      COMMON    N            , M             , A             , B
    1     ,     R            , N2            , N1            , NR
    2     ,     NM           , KFLAG         , JFLAG         , COST
    3     ,     LEVM         , NRN2          , NM1           , NN2
      COMMON    ISUCC(40,40) , LISUCC(40)    , IPRED(40,40)  , LIPRED(40)
    1     ,     INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)
    2     ,     GLEVEL(40)   , LGLIST(40)    , HLIST(40,40)  , TIME
      COMMON    T            , RTCONN(100)   , S             , RSCONN(100)
      COMMON    IFLAG        ,POINTA         ,ESS1S(40)      ,F$1(32)
```

```
     1    ,F$UB1          ,INPTCV(32)    ,LISTC(40)     ,POINTC        E1 21500
     2    ,LISTL(40)      ,POINTL        ,ORIGIN(40)    ,IPATH(40)     E1 21510
     3    ,POINTR         ,VF$1(32)      ,VF$UB1        ,GSMALL(40,32)E1 21520
      COMMON   POTAB(200,42),PPOTAB(40)  ,LPOTAB(40)    ,NRPLC(2)      E1 21530
     1    ,RPLC(2,40)     ,IDXO(32)      ,IDXOE(32)     ,IDX1(32)      E1 21540
     2    ,IDX1E(32)      ,SUMP(32)      ,SETT1(32)     ,NOT1          E1 21550
     3    ,SETS1(40)      ,NOS1          ,SETS(40)      ,NOS           E1 21560
     4    ,STS            ,SUMS2(32)     ,SETS2(200)    ,NOS2          E1 21570
     5    ,LIP            ,NODE          ,KEYA          ,KEYB          E1 21580
     6    ,NO0            ,NO1           ,NO1E          ,$GT           E1 21590
     7    ,$LTH           ,$PW           ,$NOE          ,G$$$$$        E1 21600
      COMMON              NOT1SV        ,NOS1SV        ,LMTS2          E1 21610
      DIMENSION X(40),LX(40,2),OUTO(40)                                E1 21620
C     ENTRY PRESUC                                                     E1 21630
    1 CONTINUE                                                         E1 21640
      DO 10 GI=1,NR                                                    E1 21650
       LS=0                                                            E1 21660
       LP=0                                                            E1 21670
       DO 5 GJ=1,NR                                                    E1 21680
        IF(INC$MX(GI,GJ).EQ.0) GO TO 3                                 E1 21690
         LS=LS+1                                                       E1 21700
         ISUCC(LS,GI)=GJ                                               E1 21710
         GO TO 5                                                       E1 21720
    3   IF(INC$MX(GJ,GI).EQ.0) GO TO 5                                 E1 21730
         LP=LP+1                                                       E1 21740
         IPRED(LP,GI)=GJ                                               E1 21750
    5  CONTINUE                                                        E1 21760
       LISUCC(GI)=LS                                                   E1 21770
       LIPRED(GI)=LP                                                   E1 21780
   10 CONTINUE                                                         E1 21790
C                                                                      E1 21800
      ENTRY SUCCES                                                     E1 21810
      DO 21 GI=1,NR                                                    E1 21820
      DO 21 GJ=1,NR                                                    E1 21830
       SUC$MX(GI,GJ)=0                                                 E1 21840
   21 CONTINUE                                                         E1 21850
      DO 30 GJ=N1,NR                                                   E1 21860
      DO 22 GS=1,NR                                                    E1 21870
        X(GS)=0                                                        E1 21880
   22  CONTINUE                                                        E1 21890
       X(GJ)=1                                                         E1 21900
       LO=1                                                            E1 21910
       LX(1,1)=GJ                                                      E1 21920
       V=1                                                             E1 21930
   23  CONTINUE                                                        E1 21940
       V=1-V                                                           E1 21950
       SW0=1+V                                                         E1 21960
       SW1=2-V                                                         E1 21970
       L1=0                                                            E1 21980
       DO 28 LL=1,LO                                                   E1 21990
        GM=LX(LL,SW0)                                                  E1 22000
        LIP=LIPRED(GM)                                                 E1 22010
        IF(LIP.EQ.0) GO TO 28                                          E1 22020
        DO 26 LP=1,LIP                                                 E1 22030
         GP=IPRED(LP,GM)                                               E1 22040
         IF(X(GP).GT.0) GO TO 26                                       E1 22050
          SUC$MX(GP,GJ)=1                                              E1 22060
          L1=L1+1                                                      E1 22070
          LX(L1,SW1)=GP                                                E1 22080
          X(GP)=1                                                      E1 22090
   26   CONTINUE                                                       E1 22100
   28  CONTINUE                                                          E1 22110
        IF(L1.EQ.0) GO TO 30                                             E1 22120
        L0=L1                                                            E1 22130
        GO TO 23                                                         E1 22140
   30 CONTINUE                                                           E1 22150
C                                                                        E1 22160
C     ENTRY LEVEL                                                        E1 22170
      DO 40 GJ=1,NR                                                      E1 22180
       OUTO(GJ)=LISUCC(GJ)                                               E1 22190
       GLEVEL(GJ)=-1                                                     E1 22200
   40  CONTINUE                                                          E1 22210
      LEV=0                                                              E1 22220
   45 LEV=LEV+1                                                          E1 22230
      G=0                                                                E1 22240
      DO 50 GJ=1,NR                                                      E1 22250
       IF(OUTO(GJ).GT.0 .OR. GLEVEL(GJ).GT.0) GO TO 50                   E1 22260
       G=G+1                                                             E1 22270
       HLIST(G,LEV)=GJ                                                   E1 22280
       GLEVEL(GJ)=LEV                                                    E1 22290
   50 CONTINUE                                                           E1 22300
      IF(G.EQ.0) RETURN                                                  E1 22310
      LGLIST(LEV)=G                                                      E1 22320
      DO 60 GG=1,G                                                       E1 22330
       GJ=HLIST(GG,LEV)                                                  E1 22340
       LIP=LIPRED(GJ)                                                    E1 22350
       IF(LIP.EQ.0) GO TO 60                                             E1 22360
       DO 55 LP=1,LIP                                                    E1 22370
        GP=IPRED(LP,GJ)                                                  E1 22380
        OUTO(GP)=OUTO(GP)-1                                              E1 22390
   55  CONTINUE                                                          E1 22400
   60 CONTINUE                                                           E1 22410
      LEVM=LEV                                                           E1 22420
      GO TO 45                                                           E1 22430
C                                                                        E1 22440
C                                                                        E1 22450
C                                                                        E1 22460
      ENTRY PVALUE                                                       E1 22470
      DO 100 L=NN2,NRN2                                                  E1 22480
       P$(1,L)=1                                                         E1 22490
  100 CONTINUE                                                           E1 22500
C                                                                        E1 22510
      LEV=LEVM                                                           E1 22520
  110 CONTINUE                                                           E1 22530
      LO=LGLIST(LEV)                                                     E1 22540
      DO 130 L=1,LO                                                      E1 22550
       GI=HLIST(L,LEV)                                                   E1 22560
       LIS=LISUCC(GI)                                                    E1 22570
       BSGI=(GI-1)*N2                                                    E1 22580
       LJTH=0                                                            E1 22590
       DO 115 JTH=1,N2                                                   E1 22600
        IF(P$(1,BSGI+JTH).EQ.0) GO TO 115                                E1 22610
        LJTH=LJTH+1                                                      E1 22620
        X(LJTH)=JTH                                                      E1 22630
  115  CONTINUE                                                          E1 22640
       IF(LJTH.EQ.0) GO TO 130                                           E1 22650
       DO 125 LS=1,LIS                                                   E1 22660
        GS=ISUCC(LS,GI)                                                  E1 22670
        BSGS=(GS-1)*N2                                                   E1 22680
        DO 120 LJ=1,LJTH                                                 E1 22690
         P$(1,X(LJ)+BSGS)=0                                              E1 22700
  120   CONTINUE                                                         E1 22710
  125  CONTINUE
  130 CONTINUE
      LEV=LEV-1
      IF(LEV.GE.2) GO TO 110
      RETURN
C
C
C
      ENTRY RSTRCT(KEYRST)
      KEYRST=0
      IF(LEVM.GT.LMAX)GO TO 160
      DO 150 GI=N1,NR
       IF(LIPRED(GI).GT.FANIN)GO TO 160
       IF(LISUCC(GI).GT.FANOUT)GO TO 160
  150 CONTINUE
      RETURN
  160 KEYRST=1
      RETURN
      ENTRY UNNECE
C***** THIS ENTRY DISCONNECT ALL GATES FROM WHICH THERE IS NO PATH
C       TO OUTPUT GATES *****
      TS=T
      DO 209 GI=NM1,NR
       IF(GLEVEL(GI).EQ.1) GO TO 207
       DO 205 GJ=N1,NM
        IF(SUC$MX(GI,GJ).GT.0) GO TO 209
  205  CONTINUE
C***** GI IS REDUNDANT *****
  207 CONTINUE
       LIP=LIPRED(GI)
       IF(LIP.EQ.0) GO TO 206
       DO 203 LI=1,LIP
        GK=IPRED(LI,GI)
      IF(INC$MX(GK,GI).LE.0) GO TO 203
        T=T+1
        RTCONN(T)=100*GK+GI
        INC$MX(GK,GI)=0
  203  CONTINUE
  206  LIS=LISUCC(GI)
       IF(LIS.EQ.0) GO TO 209
       DO 204 LI=1,LIS
        GK=ISUCC(LI,GI)
      IF(INC$MX(GI,GK).LE.0) GO TO 204
        T=T+1
        RTCONN(T)=100*GI+GK
        INC$MX(GI,GK)=0
  204  CONTINUE
  209 CONTINUE
      IF(T.GT.TS) GO TO 1
      RETURN
      END
```

```
C
C
C*******************************************************************************
C                                                                             *
C                                                                             *
C         PPPP    RRPR     OOO     GGG    RRPR      A     M     M             *
C         P   P   R   R   O   O   G   G   R   R    A A    MM   MM             *
C         P   P   R   R   O   O   G       R   R   A   A   M M M M             *
C         PPPP    RRRP    O   O   G  GG   PRRR    AAAAA   M  M  M             *
C         P       R  R    O   O   G   G   R  R    A   A   M     M             *
C         P       R   R    OOO     GGG    R   R   A   A   M     M             *
C                                                                             *
C                                                                             *
C                                                                             *
C N   N   EEEEE   TTTTT   TTTTT   RRRR      A             EEEEE    222        *
C NN  N   E         T       T     R   R    A A            E       2   2       *
C N N N   E         T       T     R   R   A   A           E          2        *
C N  NN   EEE       T       T     RRPR    AAAAA   XXXXX   EEE       2         *
C N   N   E         T       T     R  R    A   A           E        2          *
C N   N   EEEEE     T       T     P   R   A   A           EEEEE   22222       *
C                                                                             *
C                                                                             *
C*******************************************************************************
C
C
C     SUBROUTINE MAIN                                                   E2 00010
C     EDITION BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBE2 00020
C                                                                       E2 00030
C     NOTE: ALL COMMON VARIBLES MIGHT NOT BE USED IN THIS PROGRAM.      E2 00040
C                                                                       E2 00050
C     COMMON VARIABLES:                                                 E2 00060
C        $GT: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY   E2 00070
C             IN THIS COL. TELLS GATE WHERE FN. IS REALIZED.            E2 00080
C       $LTH: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY  E2 00090
C             IN THIS COL. TELLS HOW MANY CONNECTIONS MUST BE ADDED.    E2 00100
C       $NOE: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY  E2 00110
C             IN THIS COL. TELLS THE NUMBER OF 1-ERRORS CREATED IF THIS E2 00120
C             ROW IS USED.                                              E2 00130
C        $PW: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY  E2 00140
C             IN THIS COLUMN TELLS THE PREFERENCE WEIGHT.               E2 00150
C          A: WEIGHT FOR NO. OF GATES IN COMPUTING COST FUNCTION.       E2 00160
C          B: WEIGHT FOR NO. OF CONNECTIONS IN COMPUTING COST FUNCTION. E2 00170
C       COST: COST OF NETWORK - A MEASURE OF NETWORK SIZE.              E2 00180
C      ESSIS: RECORDS NO. OF ESSENTIAL 1'S IN EVERY INPUT TO CURRENT GCOE2 00190
C             (POSITIONS IN ESSIS CORRES. TO GATES NOT FEEDING GCO ARE  E2 00200
C             IGNORED).                                                 E2 00210
C      F$UB1: POINTS TO LAST ELEMENT IN F$1.                            E2 00220
C        F$1: LISTS (CONSECUTIVELY) POSITIONS OF DESIRABLE 1'S (FOR     E2 00230
C             COVERING) IN A CONNECTIBLE FUNCTION.                      E2 00240
C         GI: LABEL OF A PARTICULAR GATE.                               E2 00250
C     GLEVEL: GLEVEL(GI) TELLS WHICH LEVEL OF THE NETWORK GI IS IN.     E2 00260
C     GSMALL: STORES INTERMEDIATE AND FINAL CALCULATED CSPF'S.          E2 00270
C      HLIST: HLIST(I,J) GIVES NAME OF I-TH GATE (OR EX. VAR.) IN NET-  E2 00280
C             WORK LEVEL J.                                             E2 00290
C       IDXO: LIST OF 0-COORDINATES IN CSPFE OF THE GATE UNDER          E2 00300
C             CONSIDERATION.                                            E2 00310
C      IDXOE: LIST OF 0-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER    E2 00320
C             CONSIDERATION.                                            E2 00330
C       IDX1: LIST OF 1-COORDINATES IN CSPFE OF THE GATE UNDER          E2 00340
C             CONSIDERATION.                                            E2 00350
C      IDX1E: LIST OF 1-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER    E2 00360
```

```
C              CONSIDERATION.                                            E2 00370
C        IFLAG: SAME AS EYEFLG IN SUBROUTINE PROCII.                     E2 00380
C       INC$MX: INC$MX(GI,GJ)>0 MEANS THERE EXISTS A CONNECTION FROM GATE E2 00390
C              (OR EX. VAR.) GI TO GATE GJ.  INC$MX(GI,GJ)=0 IF NOT.     E2 00400
C       INPTCV: LISTS FOR EACH CORRESPONDING ENTRY OF F$1, HOW MANY INPUTSE2 00410
C              HAVE A '1' IN THE POSITION INDICATED BY F$1.              E2 00420
C        IPATH: IPATH(GI)=1 MEANS GATE GI IS ON A PATH FROM A CERTAIN GATEE2 00430
C              TO AN OUTPUT GATE.  OTHERWISE IPATH(GI) = 0.              E2 00440
C        IPRED: IPRED(I,GJ) GIVES THE NAME OF THE I-TH GATE OR EX. VAR. INE2 00450
C              A LIST OF GATES AND EX. VAR. FEEDING GJ.                  E2 00460
C        ISUCC: ISUCC(I,GJ) GIVES THE NAME OF THE I-TH GATE FED BY GJ.   E2 00470
C        JFLAG: SAME AS JAYFLG IN SUBROUTINE PROCII.                     E2 00480
C         KEYA: A FLAG INDICATING IF ANY ERROR COMPENSATION HAS BEEN     E2 00490
C              PERFORMED.                                                E2 00500
C         KEYB: A FLAG INDICATING IF ANY PRIMARY 0-ERROR-COORDINATES HAS  E2 00510
C              BEEN COMPENSATED.                                         E2 00520
C        KFLAG: SAME AS KEIFLG IN PROCII.                                E2 00530
C         LEVM: NUMBER OF LEVELS IN THE NETWORK (NOTE EX. VAR. ARE ALSO  E2 00540
C              ASSIGNED LEVELS JUST LIKE GATES).                         E2 00550
C       LGLIST: LGLIST(J) TELLS NO. OF GATES AND EX. VAR. IN LEVEL J OF  E2 00560
C              NETWORK.                                                  E2 00570
C          LIP: NUMBER OF PREDECESSORS FOR THE GATE UNDER CONSIDERATION.  E2 00580
C       LIPRED: LIPRED(GI) TELLS NO. OF IMMEDIATE PREDECESSORS OF GATE GI.E2 00590
C       LISTC: ORDERED LIST OF CONNECTIBLE INPUTS TO GCO.  ORDERED BY    E2 00600
C              DECREASING NO. OF 0'S IN GCO COVERED.                     E2 00610
C       LISTL: ORDERED LIST OF GATES AND EX. VAR. WHICH ORIGINALLY FED   E2 00620
C              GCO AND WHICH HAVE NOT YET BEEN DISCONNECTED.  ORDERED BY E2 00630
C              DECREASING NO. OF ESSENTIAL 1'S.                          E2 00640
C      LISUCC: LISUCC(GI) TELLS NO. OF IMMEDIATE SUCCESSORS OF GATE (OR  E2 00650
C              EX. VAR.) GI.                                             E2 00660
C        LMTS2: UPPER LIMIT OF THE NUMBER OF ELEMENTS IN SET S2.         E2 00670
C       LPOTAB: FOR GATE GI, LPOTAB(GI) POINTS TO LAST ROW OF POTAB      E2 00680
C              CONCERNING GI.                                            E2 00690
C            M: NUMBER OF NETWORK OUTPUT GATES.                          E2 00700
C            N: NUMBER OF EXTERNAL VARIABLES (OR INPUT FNC.) AVAILABLE.  E2 00710
C       NEPMAX: FOR ERROR COMPENSATION PROGRAMS.  IF MORE THAN NEPMAX    E2 00720
C              ERROR POSITIONS OCCUR WHEN A PARTICULAR GATE IS REMOVED,  E2 00730
C              PROGRAM SKIPS ATTEMPT TO COMPENSATE FOR THAT GATE'S       E2 00740
C              REMOVAL.  VALUE CAN BE SPECIFIED BY USER, OTHERWISE EQUAL E2 00750
C              TO ONE HALF OF N2 BY DEFAULT.                             E2 00760
C           NM: SUM OF N PLUS M                                          E2 00770
C          NM1: SUM OF NM PLUS 1.                                        E2 00780
C          NN2: PRODUCT OF N AND N2.                                     E2 00790
C          NOS: NUMBER OF ELEMENTS IN SET S.                             E2 00800
C         NOS1: NUMBER OF ELEMENTS IN SET S1.                            E2 00810
C       NOS1SV: NUMBER OF ELEMENTS IN SET S1 BEFORE ENTERING SUBROUTINE  E2 00820
C              RPLCF.                                                    E2 00830
C         NOS2: NUMBER OF ELEMENTS IN SET S2.                            E2 00840
C         NOT1: NUMBER OF ELEMENTS IN SET T1.                            E2 00850
C       NOT1SV: NUMBER OF ELEMENTS IN SET T1 BEFORE ENTERING SUBROUTINE  E2 00860
C              RPLCF.                                                    E2 00870
C          NO0: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0.                 E2 00880
C         NO0E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0E.                E2 00890
C          NO1: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1.                 E2 00900
C         NO1E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1E.                E2 00910
C           NR: SUM OF N PLUS R.                                         E2 00920
C         NRN2: PRODUCT OF NR AND N2.                                    E2 00930
C       NRPLC: NRPLC(I) STORES THE NUMBER OF ELEMENTS IN RPLC(I,*)       E2 00940
C                                                      FOR I=1,2.        E2 00950
C           N1: SUM OF N PLUS 1.                                         E2 00960
C           N2: NUMBER OF DIFFERENT INPUT COMBINATIONS TO BE CONSIDERED  E2 00970
```

```
C           (USUALLY 2 TO THE POWER N).                                         E2 00980
C    ORIGIN: ORIGIN(GI)=1 MEANS GI ORIGINALLY CONNECTED TO GCO.                 E2 00990
C            ORIGIN(GI)=0 MEANS GI DID NOT FEED GCO ORIGINALLY.                 E2 01000
C        P$: P$(1,-) CONSECUTIVELY LISTS OUTPUTS OF EVERY EX. VAR. AND          E2 01010
C            EVERY GATE (FOR EVERY INPUT COMBINATION): P$(1,1),...,             E2 01020
C            P$(1,N2) FOR FIRST EX VAR; P$(1,N2+1),...,P$(1,2*N2) FOR           E2 01030
C            SECOND EX VAR; ... ; P$(1,N*N2+1),..., P$(1,N*N2+N2) FOR           E2 01040
C            FIRST GATE; ETC.  P$(2,-) IS USED AS WORK SPACE FOR                E2 01050
C            CALCULATIONS ASSOCIATED WITH P$(1,-).                              E2 01060
C       PCO: FOR ERROR COMPENSATION PROCEDURES.  PCO IS THE GATE                E2 01070
C            REMOVED FROM ORIGINAL NETWORK TO OBTAIN CURRENT ALTERED            E2 01080
C            NETWORK.                                                           E2 01090
C    POINTA: NOT USED.                                                          E2 01100
C    POINTC: POINTS TO LAST ELEMENT IN LISTC.                                   E2 01110
C    POINTL: POINTS TO LAST ELEMENT IN LISTL.                                   E2 01120
C    POINTR: POINTS TO LAST ELEMENT IN RNEC1 (IN SUBROUTINE SUBSTI).            E2 01130
C     POTAB: POTENTIAL OUTPUT TABLE. HOLDS INFORMATION ABOUT ALL                E2 01140
C            COMBINATIONS OF CONNECTIONS TO FORM NEW (AND HOPEFULLY             E2 01150
C            USEFUL) FUNCTIONS.                                                 E2 01160
C    PPOTAB: FOR GATE GI, PPOTAB(GI) POINTS TO FIRST OF A SEQUENCE OF           E2 01170
C            ROWS OF POTAB CONCERNING GI.                                       E2 01180
C         R: NUMBER OF GATES IN THE NETWORK (EXCLUDES EX VAR, ALSO              E2 01190
C            NOTE SOME OF R GATES MAY BE ISOLATED).                             E2 01200
C      RPLC: RPLC(1,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE           E2 01210
C                      ERROR-COORDINATES OF WEIGHT 2 OR ABOVE.                  E2 01220
C            RPLC(2,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE           E2 01230
C                      AT LEAST ONE ERROR-COORDINATE OF WEIGHT 1.               E2 01240
C    RSCONN: LIST OF CONNECTIONS ADDED TO A NETWORK (IN CODED FORM).            E2 01250
C    RTCONN: LIST OF CONNECTIONS REMOVED FROM A NETWORK (CODED FORM).           E2 01260
C         S: NO. OF CONNECTIONS ADDED TO A NETWORK.  POINTS TO LAST             E2 01270
C            ENTRY IN RSCONN.                                                   E2 01280
C      SETS: SET S CONSISTING OF INPUTS OF THE GATE UNDER CONSIDERATIONE2 01290
C            WHICH ARE TO BE REPLACED IF POSSIBLE.                              E2 01300
C     SETS1: SET S1 CONSISTING OF ELEMENTS OF SET S WHICH CAN BE                E2 01310
C            REPLACED BY ELEMENTS IN SET S2.                                    E2 01320
C     SETS2: SET S2 CONSISTING OF FUNCTIONS WHICH ARE CANDIDATES FOR            E2 01330
C            REPLACING ELEMENTS IN SET S.                                       E2 01340
C     SETT1: SET T1 CONSISTING OF ESSENTIAL ONES COVERED BY ELEMENTS INE2 01350
C                                                               SET S1.         E2 01360
C       STS: STARTING ELEMENT OF SET S.                                         E2 01370
C    SUC$MX: SUC$MX(GI,GJ)>0 MEANS GATE GJ IS A SUCCESSOR OF GATE GI.           E2 01380
C            SUC$MX(GI,GJ)=0 IF NOT.                                            E2 01390
C      SUMP: SUM OF ALL ACTIVE INPUTS OF THE GATE UNDER CONSIDERATION.          E2 01400
C     SUMS2: SUM OF ALL ACTIVE ELEMENTS OF SET S2.                              E2 01410
C         T: NUMBER OF CONNECTIONS REMOVED FROM A NETWORK.  POINTS TO           E2 01420
C            LAST ENTRY IN RTCONN.                                              E2 01430
C      TIME: USED TO STORE AMOUNT OF ELAPSED COMPUTATION TIME.                  E2 01440
C     UNAME: MNEMONIC NAMES FOR EXTERNAL VARIABLES AND GATES.                   E2 01450
C    VF$UB1: POINTS TO LAST ELEMENT IN VF$1.                                    E2 01460
C      VF$1: SIMILAR TO F$1, EXCEPT THIS LISTS JUST COMPONENT POSITIONSE2 01470
C            (OF 0'S IN CSPF VECTOR OF GCO) COVERED ONLY BY REMAINING           E2 01480
C            ORIGINALLY CONNECTED INPUTS TO GCO.                                E2 01490
C                                                                               E2 01500
C                                                                               E2 01510
C                                                                               E2 01520
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                                    E2 01530
      COMMON NEPMAX                                                             E2 01540
      COMMON    N             , M             , A             , B               E2 01550
     1      ,     R           , N2            , N1            , NR              E2 01560
     2      ,     NM          , KFLAG         , JFLAG         , COST            E2 01570
     3      ,     LEVM        , NRN2          , NM1           , NN2             E2 01580
```

```
      COMMON    ISUCC(40,40) , LISUCC(40)    , IPRED(40,40) , LIPRED(40)
     1     ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)
     2     ,    GLEVEL(40)   , LGLIST(40)    , HLIST(40,40) , TIME
      COMMON    T            , RTCONN(100)   , S            , RSCONN(100)
      COMMON    IFLAG         ,POINTA        ,ESS1S(40)     ,F$1(32)
     1     ,F$UB1             ,INPTCV(32)    ,LISTC(40)     ,POINTC
     2     ,LISTL(40)         ,POINTL        ,ORIGIN(40)    ,IPATH(40)
     3     ,POINTR            ,VF$1(32)      ,VF$UB1        ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)     ,LPOTAB(40)    ,NRPLC(2)
     1     ,RPLC(2,40)        ,IDXO(32)      ,IDXOE(32)     ,IDX1(32)
     2     ,IDX1E(32)         ,SUMP(32)      ,SETT1(32)     ,NOT1
     3     ,SETS1(40)         ,NOS1          ,SETS(40)      ,NOS
     4     ,STS               ,SUMS2(32)     ,SETS2(200)    ,NOS2
     5     ,LIP               ,NOOE          ,KEYA          ,KEYB
     6     ,NOO               ,NO1           ,NO1E          ,$GT
     7     ,$LTH              ,$PW           ,$NOE          ,GI
      COMMON                  NOT1SV        ,NOS1SV        ,LMTS2
      DIMENSION INTLIS(144),UGATE(40),UHEAD(20)
      DATA KOUNT5 /0/, UBLANK/'    '/
C     NEPMAX IS THE MAXIMUM ALLOWABLE NUMBER OF ERROR POSITIONS
  990 READ(5,1000,END=500) UHEAD, N, M, R, A, B, UC, NEPMAX
 1000 FORMAT(20A4/5I4,A4,I4)
      KEYXC=0
      IF(UC.NE.UBLANK) KEYXC=1
      CALL PAGE
      CALL LINE(10)
      KOUNT5=KOUNT5+1
      PRINT 2, KOUNT5
    2 FORMAT(20X,'*** OPTIMAL NOR NETWORK ***',50X,'PROBLEM NO.= ',I4 )
      CALL LINE(4)
      PRINT 1005, UHEAD
 1005 FORMAT(25X,20A4)
      CALL LINE(4)
      PRINT 10, N,M,A,B
   10 FORMAT(30X,'NUMBER OF VARIABLES =',I4 //
     1       30X,'NUMBER OF FUNCTIONS =',I4 //
     2       30X,'COST COEFFICIENT A  =',I4//
     3       47X,                 'B  =',I4)
      CALL LINE(1)
      IF(KEYXC.NE.0) GO TO 25
      PRINT 21
   21 FORMAT(1H0,29X,'--- UNCOMPLEMENTED VARIABLES  X ---')
      GO TO 30
   25 CONTINUE
      PRINT 28
   28 FORMAT(1H0,29X,'--- BOTH COMPLEMENTED AND UNCOMPLEMENTED VARIABLES
     1 X, Y ---')
   30 CONTINUE
      CALL LINE(5)
C***** SET UP EXTERNAL VARIABLES *****
      N2=2**N
      IF(NEPMAX.EQ.0)NEPMAX = N2/2
      H=N*N2
      J=N2
      L= 1
      I=0
      DO 1011 II=1,N
       J=J/2
       L=L*2
       SN= 1
       DO 1010 LL=1,L
```

```
        SN=-SN
        V=(1+SN)/2
        DO 1009 JJ=1,J
         I=I+1
         P$(1,I)=V
      IF(KEYXC.NE.0)P$(1,I+H)=1-V
 1009   CONTINUE
 1010  CONTINUE
 1011 CONTINUE
      IF(KEYXC.NE.0) N=N+N
      N1=N+1
      NM=N+M
      NM1=NM+1
      NN2=N*N2+1
      NR=N+R
      NRN2=NR*N2
      CALL OUTPUT(INC$MX,KEYXC)
C***** READ IN NETWORK INFORMATION AND SET UP INC$MX *****
      READ 1001,   CNTLIS
 1001 FORMAT(16I5)
      DO 1115 GI=1,NR
      DO 1115 GJ=1,NR
 1115 INC$MX(GI,GJ)=0
      DO 1120 I=1,144
       ITEM=CNTLIS(I)
      IF(ITEM.EQ.0) GO TO 1119
       GI=ITEM/100
       GJ=ITEM-100*GI
       INC$MX(GI,GJ)=1
       GO TO 1120
 1119 COST=A*R+B*(I-1)
       GO TO 1130
 1120 CONTINUE
 1130 CONTINUE
      CALL SUBNET
      CALL PVALUE
      CALL LINE(4)
      PRINT 1140, COST
 1140 FORMAT(20X,' ORIGINAL NETWORK    COST=', I5)
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
C
C***** ENTRY REDUNDANCY CHECK *****
      S = 0
      T = 0
      CALL UNNECE
      GATES = M
      C = 0
      DO 4 GI =  1,NR
      C = C + LISUCC(GI)
      IF(GI.LE.NM)GOTO4
      IF(LISUCC(GI).GT.0)GATES=GATES+1
    4 CONTINUE
      NEWCST = A*GATES + B*(C)
      ITERTN = 0
    3 OLDCST = NEWCST
      ITERTN = ITERTN + 1
      PRINT 2444,ITERTN
 2444 FORMAT('1',5X,'****  BEGIN ',I3,'-TH APPLICATION OF PROCCE :
     1    *************'///////)
      T=0
      S=0
C     INITIALIZE TIMER TO 10 MINUTES
      CALL STIMEZ(60000)
      TIME = KTIMEZ(0)
C**** PROCEDURE COMPENSATE ERRORS    ************************************
      CALL PROCCE(WORKED)
C     CALL FOR ELAPSED TIME
      TIME = KTIMEZ(0) - TIME
      CALL LINE(4)
      PRINT 3915
 3916 FORMAT(20X,'TIME ELAPSED =',I8,'  CENTISECONDS')
 3915 FORMAT(20X,'NETWORK DERIVED BY PROCCE')
      PRINT 3916,TIME
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
      GATES = M
      C = 0
      DO 36 GI = 1,NR
      C = C + LISUCC(GI)
      IF(GI.LE.NM) GO TO 36
      IF(LISUCC(GI).GT.0) GATES = GATES + 1
   36 CONTINUE
      NEWCST = A*GATES + B*C
      IF(NEWCST.LT.OLDCST)GO TO 37
      PRINT 105
  105 FORMAT(1H ,10X,'NO REDUNDANCY FOUND.')
      GO TO 990
   37 CALL LINE(3)
      PRINT 320,NEWCST
  320 FORMAT(9X,'* A NETWORK DERIVED BY PROCCE'/9X,' COST=',I5,'.')
      IF(WORKED.EQ.1)GO TO 3
      CALL LINE(3)
      PRINT 301
  301 FORMAT(9X,'* PROCCE CANNOT REDUCE THE PRECEDING NETWORK FURTHER')
      GO TO 990
  500 STOP
      END
```

```
**************************************************
*                                                *
*          THE REST OF THE SUBROUTINES           *
*          REQUIRED FOR NETTRA-E2 ARE:           *
*                                                *
*  CALS1, CONECT, FORC, MINI2, ORDRQ2, OUTPUT,   *
*     POT, PROCCE, RCEC, RPLCF, AND SUBNET.      *
*                                                *
*                                                *
*    (THESE ARE IDENTICAL TO SUBROUTINES OF      *
*     THE SAME NAMES LISTED FOR NETTRA-E1.)      *
*                                                *
**************************************************
```

```
C
C
C********************************************************************************
C                                                                               *
C                                                                               *
C         PPPP    RRRR     OOO     GGG    RRRR       A    M      M              *
C         P   P   R   R   O  O    G   G   R    R    A A   MM    MM              *
C         P   P   R   R   O   O   G       R    R   A   A  M M M M               *
C         PPPP    RRRR    O   O   G  GG   RRRR     AAAAA  M  M  M               *
C         P       R  R    O   O   G   G   R   R    A   A  M     M               *
C         P       R   R    OOO     GGG    R    R   A   A  M     M               *
C                                                                               *
C                                                                               *
C                                                                               *
C N   N   EEEEE   TTTTT   TTTTT   RRRR       A             EEEEE   33333        *
C NN  N   E         T       T     R   R    A A             E          3         *
C N N N   E         T       T     R   R   A   A            E         3          *
C N  NN   EEE       T       T     RRRR    AAAAA    XXXXX   EEE        3         *
C N   N   E         T       T     R  R    A   A            E       3  3         *
C N   N   EEEEE     T       T     R   R   A   A            EEEEE    333         *
C                                                                               *
C                                                                               *
C********************************************************************************
C
C
C     SUBROUTINE MAIN
C     EDITION AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
C
C     NOTE: ALL COMMON VARIBLES MIGHT NOT BE USED IN THIS PROGRAM.
C
C     COMMON VARIABLES:
C        $GT: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY
C             IN THIS COL. TELLS GATE WHERE FN. IS REALIZED.
C       $LTH: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C             IN THIS COL. TELLS HOW MANY CONNECTIONS MUST BE ADDED.
C       $NDE: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C             IN THIS COL. TELLS THE NUMBER OF 1-ERRORS CREATED IF THIS
C             ROW IS USED.
C        $PW: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C             IN THIS COLUMN TELLS THE PREFERENCE WEIGHT.
C          A: WEIGHT FOR NO. OF GATES IN COMPUTING COST FUNCTION.
C          B: WEIGHT FOR NO. OF CONNECTIONS IN COMPUTING COST FUNCTION.
C       COST: COST OF NETWORK - A MEASURE OF NETWORK SIZE.
C      ESS1S: RECORDS NO. OF ESSENTIAL 1'S IN EVERY INPUT TO CURRENT GCO
C             (POSITIONS IN ESS1S CORRES. TO GATES NOT FEEDING GCO ARE
C             IGNORED).
C      F$UB1: POINTS TO LAST ELEMENT IN F$1.
C        F$1: LISTS (CONSECUTIVELY) POSITIONS OF DESIRABLE 1'S (FOR
C             COVERING) IN A CONNECTIBLE FUNCTION.
C         GI: LABEL OF A PARTICULAR GATE.
C     GLEVEL: GLEVEL(GI) TELLS WHICH LEVEL OF THE NETWORK GI IS IN.
C     GSMALL: STORES INTERMEDIATE AND FINAL CALCULATED CSPF'S.
C      HLIST: HLIST(I,J) GIVES NAME OF I-TH GATE (OR EX. VAR.) IN NET-
C             WORK LEVEL J.
C       IDX0: LIST OF 0-COORDINATES IN CSPFE OF THE GATE UNDER
C             CONSIDERATION.
C      IDX0E: LIST OF 0-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
C             CONSIDERATION.
C       IDX1: LIST OF 1-COORDINATES IN CSPFE OF THE GATE UNDER
C             CONSIDERATION.
C      IDX1E: LIST OF 1-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
```

```
C           CONSIDERATION.                                                  E3 00370
C     IFLAG: SAME AS EYEFLG IN SUBROUTINE PROCII.                           E3 00380
C    INC$MX: INC$MX(GI,GJ)>0 MEANS THERE EXISTS A CONNECTION FROM GATE E3 00390
C           (OR EX. VAR.) GI TO GATE GJ.  INC$MX(GI,GJ)=0 IF NOT.         E3 00400
C    INPTCV: LISTS FOR EACH CORRESPONDING ENTRY OF F$1, HOW MANY INPUTSE3 00410
C           HAVE A '1' IN THE POSITION INDICATED BY F$1.                  E3 00420
C     IPATH: IPATH(GI)=1 MEANS GATE GI IS ON A PATH FROM A CERTAIN GATEE3 00430
C           TO AN OUTPUT GATE.  OTHERWISE IPATH(GI) = 0.                  E3 00440
C     IPRED: IPRED(I,GJ) GIVES THE NAME OF THE I-TH GATE OR EX. VAR. INE3 00450
C           A LIST OF GATES AND EX. VAR. FEEDING GJ.                      E3 00460
C     ISUCC: ISUCC(I,GJ) GIVES THE NAME OF THE I-TH GATE FED BY GJ.      E3 00470
C     JFLAG: SAME AS JAYFLG IN SUBROUTINE PROCII.                         E3 00480
C      KEYA: A FLAG INDICATING IF ANY ERROR COMPENSATION HAS BEEN         E3 00490
C           PERFORMED.                                                    E3 00500
C      KEYB: A FLAG INDICATING IF ANY PRIMARY 0-ERROR-COORDINATES HAS  E3 00510
C           BEEN COMPENSATED.                                             E3 00520
C     KFLAG: SAME AS KEIFLG IN PROCII.                                    E3 00530
C      LEVM: NUMBER OF LEVELS IN THE NETWORK (NOTE EX. VAR. ARE ALSO   E3 00540
C           ASSIGNED LEVELS JUST LIKE GATES).                             E3 00550
C    LGLIST: LGLIST(J) TELLS NO. OF GATES AND EX. VAR. IN LEVEL J OF    E3 00560
C           NETWORK.                                                      E3 00570
C       LIP: NUMBER OF PREDECESSORS FOR THE GATE UNDER CONSIDERATION.  E3 00580
C    LIPRED: LIPRED(GI) TELLS NO. OF IMMEDIATE PREDECESSORS OF GATE GI.E3 00590
C     LISTC: ORDERED LIST OF CONNECTIBLE INPUTS TO GCO.  ORDERED BY     E3 00600
C           DECREASING NO. OF 0'S IN GCO COVERED.                         E3 00610
C     LISTL: ORDERED LIST OF GATES AND EX. VAR. WHICH ORIGINALLY FED    E3 00620
C           GCO AND WHICH HAVE NOT YET BEEN DISCONNECTED.  ORDERED BY  E3 00630
C           DECREASING NO. OF ESSENTIAL 1'S.                              E3 00640
C    LISUCC: LISUCC(GI) TELLS NO. OF IMMEDIATE SUCCESSORS OF GATE (OR   E3 00650
C           EX. VAR.) GI.                                                 E3 00660
C     LMTS2: UPPER LIMIT OF THE NUMBER OF ELEMENTS IN SET S2.            E3 00670
C    LPOTAB: FOR GATE GI, LPOTAB(GI) POINTS TO LAST ROW OF POTAB         E3 00680
C           CONCERNING GI.                                                E3 00690
C         M: NUMBER OF NETWORK OUTPUT GATES.                              E3 00700
C         N: NUMBER OF EXTERNAL VARIABLES (OR INPUT FNC.) AVAILABLE.     E3 00710
C    NEPMAX: FOR ERROR COMPENSATION PROGRAMS.  IF MORE THAN NEPMAX      E3 00720
C           ERROR POSITIONS OCCUR WHEN A PARTICULAR GATE IS REMOVED,   E3 00730
C           PROGRAM SKIPS ATTEMPT TO COMPENSATE FOR THAT GATE'S         E3 00740
C           REMOVAL.  VALUE CAN BE SPECIFIED BY USER, OTHERWISE EQUAL E3 00750
C           TO ONE HALF OF N2 BY DEFAULT.                                 E3 00760
C        NM: SUM OF N PLUS M                                              E3 00770
C       NM1: SUM OF NM PLUS 1.                                            E3 00780
C       NN2: PRODUCT OF N AND N2.                                         E3 00790
C       NOS: NUMBER OF ELEMENTS IN SET S.                                 E3 00800
C      NOS1: NUMBER OF ELEMENTS IN SET S1.                                E3 00810
C    NOS1SV: NUMBER OF ELEMENTS IN SET S1 BEFORE ENTERING SUBROUTINE    E3 00820
C           RPLCF.                                                        E3 00830
C      NOS2: NUMBER OF ELEMENTS IN SET S2.                                E3 00840
C      NOT1: NUMBER OF ELEMENTS IN SET T1.                                E3 00850
C    NOT1SV: NUMBER OF ELEMENTS IN SET T1 BEFORE ENTERING SUBROUTINE    E3 00860
C           RPLCF.                                                        E3 00870
C       NO0: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0.                     E3 00880
C      NO0E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0E.                    E3 00890
C       NO1: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1.                     E3 00900
C      NO1E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1E.                    E3 00910
C        NR: SUM OF N PLUS R.                                             E3 00920
C      NRN2: PRODUCT OF NR AND N2.                                        E3 00930
C     NRPLC: NRPLC(I) STORES THE NUMBER OF ELEMENTS IN RPLC(I,*)         E3 00940
C                                                      FOR I=1,2.        E3 00950
C        N1: SUM OF N PLUS 1.                                             E3 00960
C        N2: NUMBER OF DIFFERENT INPUT COMBINATIONS TO BE CONSIDERED    E3 00970
```

```
C             (USUALLY 2 TO THE POWER N).                                      E3 00980
C      ORIGIN: ORIGIN(GI)=1 MEANS GI ORIGINALLY CONNECTED TO GCO.              E3 00990
C              ORIGIN(GI)=0 MEANS GI DID NOT FEED GCO ORIGINALLY.              E3 01000
C          P$: P$(1,-) CONSECUTIVELY LISTS OUTPUTS OF EVERY EX. VAR. AND E3 01010
C              EVERY GATE (FOR EVERY INPUT COMBINATION): P$(1,1),....,         E3 01020
C              P$(1,N2) FOR FIRST EX VAR; P$(1,N2+1),...,P$(1,2*N2) FOR        E3 01030
C              SECOND EX VAR; ... ; P$(1,N*N2+1),..., P$(1,N*N2+N2) FOR        E3 01040
C              FIRST GATE; ETC.  P$(2,-) IS USED AS WORK SPACE FOR             E3 01050
C              CALCULATIONS ASSOCIATED WITH P$(1,-).                           E3 01060
C         PCO: FOR ERROR COMPENSATION PROCEDURES.  PCO IS THE GATE             E3 01070
C              REMOVED FROM ORIGINAL NETWORK TO OBTAIN CURRENT ALTERED         E3 01080
C              NETWORK.                                                        E3 01090
C      POINTA: NOT USED.                                                       E3 01100
C      POINTC: POINTS TO LAST ELEMENT IN LISTC.                                E3 01110
C      POINTL: POINTS TO LAST ELEMENT IN LISTL.                                E3 01120
C      POINTR: POINTS TO LAST ELEMENT IN RNEC1 (IN SUBROUTINE SUBSTI).         E3 01130
C       POTAB: POTENTIAL OUTPUT TABLE.  HOLDS INFORMATION ABOUT ALL            E3 01140
C              COMBINATIONS OF CONNECTIONS TO FORM NEW (AND HOPEFULLY          E3 01150
C              USEFUL) FUNCTIONS.                                              E3 01160
C      PPOTAB: FOR GATE GI, PPOTAB(GI) POINTS TO FIRST OF A SEQUENCE OF        E3 01170
C              ROWS OF POTAB CONCERNING GI.                                    E3 01180
C           R: NUMBER OF GATES IN THE NETWORK (EXCLUDES EX VAR, ALSO           E3 01190
C              NOTE SOME OF R GATES MAY BE ISOLATED).                          E3 01200
C        RPLC: RPLC(1,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE        E3 01210
C                        ERROR-COORDINATES OF WEIGHT 2 OR ABOVE.               E3 01220
C              RPLC(2,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE        E3 01230
C                        AT LEAST ONE ERROR-COORDINATE OF WEIGHT 1.            E3 01240
C      RSCONN: LIST OF CONNECTIONS ADDED TO A NETWORK (IN CODED FORM).         E3 01250
C      RTCONN: LIST OF CONNECTIONS REMOVED FROM A NETWORK (CODED FORM).        E3 01260
C           S: NO. OF CONNECTIONS ADDED TO A NETWORK.  POINTS TO LAST          E3 01270
C              ENTRY IN RSCONN.                                                E3 01280
C        SETS: SET S CONSISTING OF INPUTS OF THE GATE UNDER CONSIDERATIONE3 01290
C              WHICH ARE TO BE REPLACED IF POSSIBLE.                           E3 01300
C       SETS1: SET S1 CONSISTING OF ELEMENTS OF SET S WHICH CAN BE             E3 01310
C              REPLACED BY ELEMENTS IN SET S2.                                 E3 01320
C       SETS2: SET S2 CONSISTING OF FUNCTIONS WHICH ARE CANDIDATES FOR         E3 01330
C              REPLACING ELEMENTS IN SET S.                                    E3 01340
C       SETT1: SET T1 CONSISTING OF ESSENTIAL ONES COVERED BY ELEMENTS INE3 01350
C                                                               SET S1.        E3 01360
C         STS: STARTING ELEMENT OF SET S.                                      E3 01370
C      SUC$MX: SUC$MX(GI,GJ)>0 MEANS GATE GJ IS A SUCCESSOR OF GATE GI.        E3 01380
C              SUC$MX(GI,GJ)=0 IF NOT.                                         E3 01390
C        SUMP: SUM OF ALL ACTIVE INPUTS OF THE GATE UNDER CONSIDERATION.       E3 01400
C       SUMS2: SUM OF ALL ACTIVE ELEMENTS OF SET S2.                           E3 01410
C           T: NUMBER OF CONNECTIONS REMOVED FROM A NETWORK.  POINTS TO        E3 01420
C              LAST ENTRY IN RTCONN.                                           E3 01430
C        TIME: USED TO STORE AMOUNT OF ELAPSED COMPUTATION TIME.               E3 01440
C       UNAME: MNEMONIC NAMES FOR EXTERNAL VARIABLES AND GATES.                E3 01450
C     VF$UB1: POINTS TO LAST ELEMENT IN VF$1.                                  E3 01460
C       VF$1: SIMILAR TO F$1, EXCEPT THIS LISTS JUST COMPONENT POSITIONSE3 01470
C              (OF 0'S IN CSPF VECTOR OF GCO) COVERED ONLY BY REMAINING        E3 01480
C              ORIGINALLY CONNECTED INPUTS TO GCO.                             E3 01490
C                                                                              E3 01500
C                                                                              E3 01510
C                                                                              E3 01520
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                                   E3 01530
      COMMON NERMAX                                                            E3 01540
      COMMON    N             , M             , A             , B              E3 01550
     1    ,     R             , N2            , N1            , NR             E3 01560
     2    ,     NM            , KFLAG         , JFLAG         , COST           E3 01570
     3    ,     LEVM          , NRN2          , NM1           , NN2            E3 01580
```

```
      COMMON    ISUCC(40,40) , LISUCC(40)    , IPRED(40,40) , LIPRED(40)
     1     ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)     , UNAME(40)
     2     ,    GLEVEL(40)   , LGLIST(40)    , HLIST(40,40) , TIME
      COMMON    T              , RTCONN(100)   , S             , RSCONN(100)
      COMMON    IFLAG          ,POINTA         ,ESS1S(40)      ,F$1(32)
     1     ,F$UB1              ,INPTCV(32)     ,LISTC(40)      ,POINTC
     2     ,LISTL(40)          ,POINTL         ,ORIGIN(40)     ,IPATH(40)
     3     ,POINTR             ,VF$1(32)       ,VF$UB1         ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)       ,LPOTAB(40)     ,NRPLC(2)
     1     ,RPLC(2,40)         ,IDX0(32)       ,IDX0E(32)      ,IDX1(32)
     2     ,IDX1E(32)          ,SUMP(32)       ,SETT1(32)      ,NOT1
     3     ,SETS1(40)          ,NOS1           ,SETS(40)       ,NOS
     4     ,STS                ,SUMS2(32)      ,SETS2(200)     ,NOS2
     5     ,LIP                ,NODF           ,KEYA           ,KEYB
     6     ,NO0                ,NO1            ,NO1E           ,$GT
     7     ,$LTH               ,$PW            ,$NOE           ,GI
      COMMON                   NOT1SV          ,NOS1SV         ,LMTS2
      DIMENSION CNTLIS(144),UGATE(40),UHEAD(20)
      DATA KOUNT5 /0/, UBLANK/'    '/
  990 READ(5,1000,END=500) UHEAD, N, M, R, A, B, UC, NEPMAX
C     NEPMAX IS THE MAXIMUM ALLOWABLE NUMBER OF ERROR POSITIONS
 1000 FORMAT(20A4/5I4,A4,I4)
      KEYXC=0
      IF(UC.NE.UBLANK) KEYXC=1
      CALL PAGE
      CALL LINE(10)
      KOUNT5=KOUNT5+1
      PRINT 2, KOUNT5
    2 FORMAT(20X,'*** OPTIMAL NOR NETWORK ***',50X,'PROBLEM NO.= ',I4 )
      CALL LINE(4)
      PRINT 1005, UHEAD
 1005 FORMAT(25X,20A4)
      CALL LINE(4)
      PRINT 10, N,M,A,B
   10 FORMAT(30X,'NUMBER OF VARIABLES =',I4 //
     1        30X,'NUMBER OF FUNCTIONS =',I4 //
     2        30X,'COST COEFFICIENT A  =',I4//
     3        47X,                'B  =',I4)
      CALL LINE(1)
      IF(KEYXC.NE.0) GO TO 25
      PRINT 21
   21 FORMAT(1H0,29X,'--- UNCOMPLEMENTED VARIABLES  X ---')
      GO TO 30
   25 CONTINUE
      PRINT 28
   28 FORMAT(1H0,29X,'--- BOTH COMPLEMENTED AND UNCOMPLEMENTED VARIABLES
     1 X, Y ---')
   30 CONTINUE
      CALL LINE(5)
C***** SET UP EXTERNAL VARIABLES *****
      N2=2**N
      IF(NEPMAX.EQ.0)NEPMAX = N2/2
      H=N*N2
      J=N2
      L= 1
      I=0
      DO 1011 II=1,N
       J=J/2
       L=L*2
       SN= 1
       DO 1010 LL=1,L
```

```
        SN=-SN
        V=(1+SN)/2
        DO 1009 JJ=1,J
         I=I+1
         P$(1,I)=V
      IF(KEYXC.NE.0)P$(1,I+H)=1-V
 1009  CONTINUE
 1010  CONTINUE
 1011 CONTINUE
      IF(KEYXC.NE.0) N=N+N
      N1=N+1
      NM=N+M
      NM1=NM+1
      NN2=N*N2+1
      NR=N+R
      NRN2=NR*N2
      CALL OUTPUT(INC$MX,KEYXC)
C***** READ IN NETWORK INFORMATION AND SET UP INC$MX *****
      READ 1001,    CNTLIS
 1001 FORMAT(16I5)
      DO 1115 GI=1,NR
      DO 1115 GJ=1,NR
 1115 INC$MX(GI,GJ)=0
      DO 1120 I=1,144
       ITEM=CNTLIS(I)
      IF(ITEM.EQ.0) GO TO 1119
       GI=ITEM/100
       GJ=ITEM-100*GI
       INC$MX(GI,GJ)=1
       GO TO 1120
 1119 COST=A*R+B*(I-1)
       GO TO 1130
 1120 CONTINUE
 1130 CONTINUE
      CALL SUBNET
      CALL PVALUE
      CALL LINE(4)
      PRINT 1140, COST
 1140 FORMAT(20X,' ORIGINAL NETWORK    COST=', I5)
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
C
C***** ENTRY REDUNDANCY CHECK *****
      T=0
      S=0
      CALL UNNECE
C     INITIALIZE TIMER TO 10 MINUTES
      CALL STIMEZ(60000)
      TIME = KTIMEZ(0)
C**** PROCEDURE COMPENSATE ERRORS   ************************************
      CALL ALPATH
C     CALL FOR ELAPSED TIME
      TIME = KTIMEZ(0) - TIME
      CALL LINE(4)
      PRINT 3915
 3916 FORMAT(20X,'TIME ELAPSED =',I8,'  CENTISECONDS')
 3915 FORMAT(20X,'NETWORKS DERIVED BY ALL-PATH PROCCE')
      PRINT 3916,TIME
      GO TO 990
```

```
C**** FOR THIS SPECIAL ALL-PATH VERSION OF MAIN, THE REST OF THE PROGRAM
C     ST.S ARE BYPASSED
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
C***** PRINT OUT NETWORK DERIVED BY REDUNDANCY CHECK *****
      IF(T.GT.0) GO TO 110
      CALL LINE(2)
  102 PRINT 105
  105 FORMAT(1H ,10X,'NO REDUNDANCY FOUND.')
      GO TO 990
C
  110 CONTINUE
C***** PRINT OUT REDUNDANT GATES *****
      CALL UNNECE
      G=0
  115 KEY=0
      DO 125 GJ=NM1,NR
       IF(LISUCC(GJ).GT.0) GO TO 125
        G=G+1
        UGATE(G)=UNAME(GJ)
        LISUCC(GJ)=9999
        LIP=LIPRED(GJ)
       IF(LIP.EQ.0) GO TO 125
        DO 120 LP=1,LIP
         GP=IPRED(LP,GJ)
         T=T+1
         INC$MX(GP,GJ)=0
         RTCONN(T)=100*GP+GJ
         LISUCC(GP)=LISUCC(GP)-1
         KEY=1
  120   CONTINUE
  125 CONTINUE
      IF(KEY.GT.0) GO TO 115
      CALL SUBNET
      CALL PVALUE
      CALL LINE(3)
  301 PRINT 302
  302 FORMAT(1H ,10X,'THE FOLLOWING RECONFIGUFATION DONE.'//)
      IF(G.EQ.0) GO TO 310
      PRINT 303,( UGATE(GG),GG=1,G)
  303 FORMAT(1H ,15X,'REDUNDANT GATE(S)'//20X,10(3X,A3))
      CALL LINE(2)
C***** PRINT OUT REMOVED AND ADDED CONNECTIONS *****
  310 IF(T.EQ.0)GOTO401
      PRINT 311
  311 FORMAT(1H ,15X,'REMOVED CONNECTION(S)')
      DO 315 TT=1,T
       ITEM=RTCONN(TT)
       GI=ITEM/100
       GJ=ITEM-GI*100
       UI=UNAME(GI)
       UJ=UNAME(GJ)
       PRINT 314,UI,UJ
  314 FORMAT(1H0,19X,'(',2X,A3,',',2X,A3,')')
  315 CONTINUE
  401 IF(S.EQ.0) GO TO 319
      CALL LINE(2)
      PRINT 316
  316 FORMAT(1H ,15X,'ADDED CONNECTION(S)')
```

```
      DO 318 SS=1,S                                                     E3 03420
       ITEM=RSCONN(SS)                                                  E3 03430
       GI=ITEM/100                                                      E3 03440
       GJ=ITEM-GI*100                                                   E3 03450
       UI=UNAME(GI)                                                     E3 03460
       UJ=UNAME(GJ)                                                     E3 03470
       PRINT 314,UI,UJ                                                  E3 03480
C                                                                       E3 03490
  318 CONTINUE                                                          E3 03500
  319 CONTINUE                                                          E3 03510
      COUNTC = 0                                                        E3 03520
      DO 6447 I = 1,NR                                                  E3 03530
 6447 COUNTC = COUNTC + LISUCC(I)                                       E3 03540
      CUNEW = A * (R - G) + B * (COUNTC)                                E3 03550
      COST = CUNEW                                                      E3 03560
      CALL LINE(3)                                                      E3 03570
      PRINT 320,    CUNEW                                               E3 03580
  320 FORMAT(9X,'* A NETWORK DERIVED BY PROCCE'/9X,' COST=',I5,'.')     E3 03590
      NEWCST = CUNEW                                                    E3 03600
C     IF(NEWCST.LT.OLDCST)GO TO 3                                       E3 03610
      GO TO 990                                                         E3 03620
  500 STOP                                                              E3 03630
      END                                                               E3 03640


      SUBROUTINE PROCCE(WORKED)                                         E3 03650
C        PROCCE  FOR  MULTI-PATH  PROGRAM  ****                         E3 03660
C     EDITION AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE3 03670
C     IF PROCCE SUCCESSFULLY COMPENSATES ERRORS, 'WORKED' IS SET TO 1, OE3 03680
C     'WORKED' IS SET TO 0                                              E3 03690
C                                                                       E3 03700
C     DEFS. OF MOST  'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.   E3 03710
C                                                                       E3 03720
C     SPECIAL COMMON VARIABLES:                                         E3 03730
C                                                                       E3 03740
C     INC$AV: STACK CONTAINING BLOCKS. EACH BLOCK CONTAINS A NETWORK'S  E3 03750
C             NAME, PARENT, COST, AND LIST OF CONNECTIONS.              E3 03760
C       NAME: NAME OF NETWORK UNDER CONSIDERATION.                      E3 03770
C     NETLST: LIST (STACK) OF POINTERS TO TOP OF BLOCKS IN INC$AV.      E3 03780
C     NLSTPT: POINTS TO TOP OF STACK NETLST.                            E3 03790
C     NTCNTR: NETWORK COUNTER - COUNTS NUMBER OF NETWORKS GENERATED SO  E3 03800
C             FAR.                                                      E3 03810
C     NTCOST: COST OF A PARTICULAR NETWORK.                             E3 03820
C     PARENT: NAME OF NETWORK FROM WHICH NETWORK 'NAME' WAS DERIVED.    E3 03830
C      $AVPT: POINTS TO FIRST FREE LOCATION IN STACK INC$AV.            E3 03840
C                                                                       E3 03850
C     VARIABLE DEFINITIONS:                                             E3 03860
C         EP: EP(I)=1 MEANS AT LEAST ONE NETWORK OUTPUT GATE HAS AN     E3 03870
C             ERRONEOUS OUTPUT IN THE I-TH COMPONENT WHEN PCO IS REMOVEDE3 03880
C             FROM THE NETWORK.  EP(I)=0 OTHERWISE.                     E3 03890
C     ERRORS: TOTAL NO. OF ERRORS IN NETWORK OUTPUTS WHEN PCO REMOVED.  E3 03900
C      GATES: NUMBER OF GATES REMOVED FROM NETWORK BY CALL TO MINI2.    E3 03910
C     IMPROV: A PARAMETER RETURNED BY MINI2.  '=1' MEANS MINI2 WAS ABLE E3 03920
C             TO REDUCE COST OF NETWORK.                                E3 03930
C        MAX: MAXIMUM NUMBER OF REQUIRED  1'S IN A CSPF VECTOR (AFTER   E3 03940
C             CALLING MINI2) PLUS 1.                                    E3 03950
C        MIN: ORIGINALLY SET TO ZERO, MIN IS INCREMENTED EACH TIME BY 1 E3 03960
C             UNTIL ITS VALUE EQUALS MAX.                               E3 03970
C        NEP: NO. OF ERROR POSITIONS FOR A GIVEN NETWORK AFTER A SE-    E3 03980
C             LECTED GATE HAS BEEN REMOVED.  AN ERROR POSITION IS A     E3 03990
C             COMPONENT POSITION WHICH IS IN ERROR FOR AT LEAST ONE     E3 04000
```

```
C            OUTPUT.                                                     E3 04010
C     NEPMAX: READ FROM INPUT CARDS, THIS PARAMETER IS PASSED TO PROCCE E3 04020
C            WHEN IT IS CALLED BY MAIN.  IT REPRESENTS THE MAXIMUM       E3 04030
C            ALLOWABLE NUMBER OF ERROR POSITIONS.  IF AN ALTERED (I.E.,E3 04040
C            SOME PCO REMOVED) NETWORK EXCEEDS THIS MAXIMUM, ERROR       E3 04050
C            COMPENSATION IS NOT ATTEMPTED FOR THAT NETWORK.             E3 04060
C     NETOUT: STORES OUTPUTS OF GATES IN ALTERED (PCO REMOVED) NETWORK. E3 04070
C     ONECNT: USED IN COUNTING NO. OF 1'S IN CSPF VECTOR OF A GATE.     E3 04080
C       ONES: AFTER THE INITIAL CALCULATION OF THE CSPF SETS IN THE      E3 04090
C            BEGINNING, ONES(GI) GIVES THE NUMBER OF 1'S IN THE CSPF     E3 04100
C            VECTOR OF GI.  THIS INFORMATION IS REQUIRED FOR GENERATINGE3 04110
C            PORDER.                                                     E3 04120
C     ORGOUT: USED TO STORE ORIGINAL (UNALTERED) NETWORK OUTPUTS IN      E3 04130
C            CODED FORM (SAME CODE AS IN GSMALL) AND (40,32) FORMAT.     E3 04140
C        PCO: CURRENT GATE REMOVED FROM ORIGINAL NETWORK TO OBTAIN       E3 04150
C            CURRENT ALTERED NETWORK.  PCO = PORDER(PCCUNT).             E3 04160
C     PCOUNT: A POINTER TO PORDER.                                       E3 04170
C     PORDER: ORDERING OF GATES ACCORDING TO NUMBER OF 1'S IN THEIR      E3 04180
C            CSPF VECTORS.  GATES ARE INDIVIDUALLY REMOVED FROM ORIGI-  E3 04190
C            NAL NETWORK IN THIS ORDER                                   E3 04200
C       PSUB: USED AS A POINTER TO PORDER DURING ITS INITIALIZATION.    E3 04210
C     OINC$M: STORES A COPY OF INC$MX FOR THE ORIGINAL NETWORK.         E3 04220
C      START: POINTS TO BEGINNING OF LIST OF NETWORK OUTPUTS IN P$.     E3 04230
C       STOP: POINTS TO END OF LIST OF NETWORK OUTPUTS IN P$.           E3 04240
C                                                                        E3 04250
C     I,J,NI,X,Y ARE USED AS JUST TEMPORARY VARIABLES.                   E3 04260
C                                                                        E3 04270
C     HOW TO INCREASE CAPACITY OF SUBROUTINE.                            E3 04280
C     DIMENSION: PORDER(X)                                               E3 04290
C                ONES(X)                                                 E3 04300
C                OINC$M(X,X) - X EQUAL TO MAX NO. OF GATES PLUS EX. VAR.E3 04310
C                EP(Y)       - Y EQUAL TO: 2**(MAX ALLOWED NO OF EX VAR)E3 04320
C                NETOUT(X,Y)                                             E3 04330
C                ORGOUT(X,Y) - X,Y AS ABOVE                              E3 04340
C                                                                        E3 04350
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)                             E3 04360
      COMMON NEPMAX                                                      E3 04370
      COMMON    N             , M             , A             , B        E3 04380
     1     ,    R             , N2            , N1            , NR       E3 04390
     2     ,    NM            , KFLAG         , JFLAG         , COST     E3 04400
     3     ,    LEVM          , NRN2          , NM1           , NN2      E3 04410
      COMMON    ISUCC(40,40)  , LISUCC(40)    , IPRED(40,40)  , LIPRED(40)  E3 04420
     1     ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)   E3 04430
     2     ,    GLEVEL(40)    , LGLIST(40)    , HLIST(40,40)  , TIME     E3 04440
      COMMON    T             , RTCONN(100)   , S             , RSCONN(100) E3 04450
      COMMON    IFLAG         ,POINTA         ,ESS1S(40)      ,F$1(32)   E3 04460
     1     ,F$UB1             ,INPTCV(32)     ,LISTC(40)      ,POINTC    E3 04470
     2     ,LISTL(40)         ,POINTL         ,ORIGIN(40)     ,IPATH(40) E3 04480
     3     ,POINTR            ,VF$1(32)       ,VF$UB1         ,GSMALL(40,32)E3 04490
      COMMON    POTAB(200,42),PPOTAB(40)      ,LPOTAB(40)     ,NRPLC(2)  E3 04500
     1     ,RPLC(2,40)        ,IDX0(32)       ,IDX0E(32)      ,IDX1(32)  E3 04510
     2     ,IDX1E(32)         ,SUMP(32)       ,SETT1(32)      ,NOT1      E3 04520
     3     ,SETS1(40)         ,NOS1           ,SETS(40)       ,NOS       E3 04530
     4     ,STS               ,SUMS2(32)      ,SETS2(200)     ,NOS2      E3 04540
     5     ,LIP               ,NO0E           ,KEYA           ,KEYB      E3 04550
     6     ,NO0               ,NO1            ,NO1E           ,$GT       E3 04560
     7     ,$LTH              ,$PW            ,$NOE           ,GI        E3 04570
      COMMON                  NOT1SV         ,NOS1SV         ,LMTS2     E3 04580
      COMMON    NTCNTR        ,PARENT         ,NAME           ,INC$AV(10000)E3 04590
     1     ,$AVPT             ,NETLST(500)    ,NLSTPT         ,NTCOST    E3 04600
      DIMENSION PORDER(40),ONES(40),OINC$M(40,40),NETOUT(40,32),        E3 04610
     1 FP(32),ORGOUT(40,32)
C      THIS SUBROUTINE ASSUMES ALL ARRAYS ARE UPDATED
C      PREVIOUS TO BEING CALLED
C
       $GT   = 33
       $LTH  = 34
       $PW   = 41
       $NDE  = 42
       WORKED = 0
       S = 0
       T = 0
       $PSAVE = $AVPT
C
C      BLOCK  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B  B
C
       CALL MINI2(IMPROV)
C      IN THIS CALL TO MINI2, GORDER WILL BE CALCULATED.  GORDER WILL BE
C      LATER IN EACH CALL TO INITGS (AN ENTRY POINT OF MINI2). NOTE THAT
C      IS NOT AFFECTED BY THE REMOVAL OF GATES FROM THE ORIGINAL NETWORK.
       IF(IMPROV.EQ.0)GO TO 1
       GAFTER = M
       C = 0
       DO 2 I = 1,NR
       C = C + LISUCC(I)
       IF(I.LE.NM)GOTO2
       IF(LISUCC(I).GT.0) GAFTER = GAFTER + 1
     2 CONTINUE
       GBEFOR = (NTCOST-8*(C+T))/A
       GATES = R - GAFTER
       PRINT 4,GATES,T
     4 FORMAT(' ',I5,' GATES AND',I3,' CONNECTIONS HAVE BEEN REMOVED FROM
      1 THE NETWORK DURING THE INITIAL CALCULATION OF THE CSPF SET')
     1 CONTINUE
C      COUNT THE NUMBER OF 1'S IN THE CSPF VECTOR FOR EACH GATE
       MAX = 0
       DO 5 I = N1,NR
       ONECNT = 0
       DO 6 J = 1,N2
       IF(GSMALL(I,J).LE.0)GO TO 6
       ONECNT = ONECNT + 1
     6 CONTINUE
       IF(ONECNT.GT.MAX)MAX=ONECNT
       ONES(I) = ONECNT
     5 CONTINUE
       MAX = MAX + 1
       MIN = -1
       PSUB = 1
     7 MIN = MIN + 1
       IF(MIN.EQ.MAX) GO TO 8
       DO 9 I = N1,NR
       IF(ONES(I).NE.MIN)GO TO 9
       PORDER(PSUB) = I
       PSUB = PSUB + 1
     9 CONTINUE
       GOTO7
     8 CONTINUE
C      SAVE ORIGINAL NETWORK
       DO 10 I = 1,NR
       DO 10 J = 1,NR
       OINC$M(I,J) = INC$MX(I,J)
    10 CONTINUE
```

```
C     SAVE ORIGINAL OUTPUTS                                             E3 05230
C     SAVE ORIGINAL OUTPUTS IN (2,1280) FORMAT                          E3 05240
      START = (N*N2) + 1                                                E3 05250
      STOP = (NM*N2)                                                    E3 05260
      DO 13 I = START, STOP                                             E3 05270
      P$(2,I) = P$(1,I)                                                 E3 05280
   13 CONTINUE                                                          E3 05290
C     SAVE ORIGINAL OUTPUTS IN CODED (40,32) FORMAT                     E3 05300
      DO 27 I = N1,NM                                                   E3 05310
      X = (   I-1) * N2                                                 E3 05320
      DO 28 J = 1,N2                                                    E3 05330
      Y = P$(1,X+J)                                                     E3 05340
      IF(Y)30,31,32                                                     E3 05350
C     COMPONENT IS DON'T CARE  (I.E., -1)                               E3 05360
   30 ORGOUT(I,J) = 0                                                   E3 05370
      GOTO 28                                                           E3 05380
C     COMPONENT IS LOGICAL ZERO                                         E3 05390
   31 ORGOUT(I,J) = -100                                                E3 05400
      GO TO 28                                                          E3 05410
C     COMPONENT IS LOGICAL ONE                                          E3 05420
   32 ORGOUT(I,J) = 1                                                   E3 05430
   28 CONTINUE                                                          E3 05440
   27 CONTINUE                                                          E3 05450
C                                                                       E3 05460
C     BLOCK  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C E3 05470
C                                                                       E3 05480
      PCOUNT = 0                                                        E3 05490
   11 PCOUNT = PCOUNT + 1                                               E3 05500
      IF(PCOUNT.GT. R)GOTO42                                            E3 05510
      PCO = PORDER(PCOUNT)                                              E3 05520
      IF(ONES(PCO).EQ.0)GO TO 11                                        E3 05530
      IF(PCO.LE.NM)GO TO 11                                             E3 05540
C     ERRORS UNCORRECTABLE, RESTORE NETWORK, TRY AGAIN                  E3 05550
      DO 19 I = 1,NR                                                    E3 05560
      DO 19 J = 1,NR                                                    E3 05570
      INC$MX(I,J) = OINC$M(I,J)                                         E3 05580
   19 CONTINUE                                                          E3 05590
C     REMOVE GATE PCO FROM THE NETWORK                                  E3 05600
      DO 12 I = 1,NR                                                    E3 05610
      IF(INC$MX(I,PCO).EQ.0)GO TO 34                                    E3 05620
      INC$MX(I,PCO) = 0                                                 E3 05630
   34 IF(INC$MX(PCO,I).EQ.0) GO TO 12                                   E3 05640
      INC$MX(PCO,I) = 0                                                 E3 05650
   12 CONTINUE                                                          E3 05660
C     UPDATE GATE OUTPUTS FOR ALTERED NETWORK                           E3 05670
C                                                                       E3 05680
C     BLOCK  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D  D E3 05690
C                                                                       E3 05700
   33 CALL SUBNET                                                       E3 05710
      CALL PVALUE                                                       E3 05720
      CALL UNNECE                                                       E3 05730
      S = 0                                                             E3 05740
      T = 0                                                             E3 05750
C     RESTORE GSMALL FOR OUTPUT GATES                                   E3 05760
      DO 29 I = N1,NM                                                   E3 05770
      DO 29 J = 1, N2                                                   E3 05780
      GSMALL(I,J) = ORGOUT(I,J)                                         E3 05790
   29 CONTINUE                                                          E3 05800
      ERRORS = 0                                                        E3 05810
      DO 24 I=1,N2                                                      E3 05820
   24 EP(I) = 0                                                         E3 05830
```

```
      DO 14 I = 1,M                                                     E3 05840
      NI = N + I                                                        E3 05850
      X = (NI - 1) * N2                                                 E3 05860
      DO 15 J = 1,N2                                                    E3 05870
      IF(GSMALL(NI,J))16,15,17                                          E3 05880
C     CASE WHERE REQUIREMENT IS A ZERO                                  E3 05890
   16 IF(P$(1,X+J).EQ.0)GO TO 15                                        E3 05900
C     CASE OF ONE WITH ERROR                                            E3 05910
      GSMALL(NI,J) = 1001                                               E3 05920
      ERRORS = ERRORS + 1                                               E3 05930
      EP(J) = 1                                                         E3 05940
      GO TO 15                                                          E3 05950
C     CASE WHERE REQUIREMENT IS A ONE                                   E3 05960
   17 IF(P$(1,X+J).EQ.1)GO TO 15                                        E3 05970
C     CASE OF ZERO WITH ERROR                                           E3 05980
      GSMALL(NI,J) = -1100                                              E3 05990
      ERRORS = ERRORS + 1                                               E3 06000
      EP(J) = 1                                                         E3 06010
   15 CONTINUE                                                          E3 06020
   14 CONTINUE                                                          E3 06030
      IF(ERRORS.EQ.0)WORKED = 1                                         E3 06040
      IF(ERRORS.EQ.0) GO TO 23                                          E3 06050
      NEP = 0                                                           E3 06060
      DO 25 I = 1,N2                                                    E3 06070
      IF(EP(I).EQ.0) GO TO 25                                           E3 06080
      NEP = NEP + 1                                                     E3 06090
   25 CONTINUE                                                          E3 06100
      IF(NEP.GT.NEPMAX) GO TO 11                                        E3 06110
C                                                                       E3 06120
C     BLOCK  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E  E E3 06130
C                                                                       E3 06140
      CALL POT                                                          E3 06150
C     'POT' IS A SUBROUTINE THAT GENERATES THE POTENTIAL OUTPUT TABLE   E3 06160
C                                                                       E3 06170
C     BLOCK  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F  F E3 06180
C                                                                       E3 06190
C     SAVE NEW NETWORK OUTPUTS                                          E3 06200
      DO 18 J = 1,N2                                                    E3 06210
      DO 18 I = N1,NM                                                   E3 06220
      NETOUT(I,J) = GSMALL(I,J)                                         E3 06230
   18 CONTINUE                                                          E3 06240
      CALL FORMGO                                                       E3 06250
      CALL INITGS                                                       E3 06260
      CALL RSEC(&11,&33)                                                E3 06270
C     NEW NETWORK HAS BEEN FOUND, PUT IT IN STACK                       E3 06280
   23 NTCNTR = NTCNTR + 1                                               E3 06290
      TEMP = $AVPT                                                      E3 06300
      DO 26 I=1,NR                                                      E3 06310
      DO 26 J=N1,NR                                                     E3 06320
      IF(INC$MX(I,J).LE.0)GO TO 26                                      E3 06330
      X = 1000*I + J                                                    E3 06340
      INC$AV($AVPT) = X                                                 E3 06350
      $AVPT = $AVPT + 1                                                 E3 06360
   26 CONTINUE                                                          E3 06370
      NLSTPT = NLSTPT + 1                                               E3 06380
      NETLST(NLSTPT) = $AVPT + 2                                        E3 06390
C     INSERT COST OF NETWORK                                            E3 06400
      GATES = M                                                         E3 06410
      DO 41 GATEI = NM1,NR                                              E3 06420
      IF(LISUCC(GATEI).GT.0)GATES=GATES+1                               E3 06430
   41 CONTINUE                                                          E3 06440
```

```
      C = $AVPT - TEMP
      CST = A*GATES + B*C
      INC$AV($AVPT) = CST
C     INSERT PARENT'S NAME
      INC$AV($AVPT+1) = NAME
C     INSERT THIS NEW NETWORK'S NAME INTO STACK
      INC$AV($AVPT+2) = NTCNTR
      $AVPT = $AVPT + 3
      PRINT 2345,NTCNTR,NAME
 2345 FORMAT('1',19X,'NETWORK NUMBER ',I4,' DERIVED BY PROCCE.  THE PARE
     1NT OF THIS NETWORK IS NUMBER ',I4////)
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
      S = 0
      T = 0
      CALL LINE(4)
      PRINT 2346,CST
 2346 FORMAT(20X,'THIS NETWORK HAS A COST OF:',I5)
      GO TO 11
   42 IF(IMPROV.EQ.0.OR.$PSAVE.NE.$AVPT)RETURN
      IF(GBEFOR.EQ.GAFTER)RETURN
C
C     IF HERE, AN IMPROVED NETWORK OF FEWER GATES HAS BEEN FOUND, BUT ON
C     THE OPERATION OF MINI2 (I.E., NOT BY RCEC).  SO NEW NETWORK WILL B
C     PRINTED OUT, BUT NOT STORED IN THE STACK.
C
C     RESTORE NETWORK DERIVED BY MINI2
      DO 43 I=1,NR
      DO 43 J=1,NR
      INC$MX(I,J) = QINC$M(I,J)
   43 CONTINUE
      CALL SUBNET
      CALL PVALUE
      NTCNTR = NTCNTR + 1
      PRINT 2348,NTCNTR,NAME
 2348 FORMAT('1',19X,'NETWORK NUMBER ',I4,' DERIVED BY MINI2.  THE PAREN
     1T OF THIS NETWORK IS NUMBER ',I4////)
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
      CST = A*GAFTER + B*C
      PRINT 2346,CST
      RETURN
      END


      SUBROUTINE ALPATH
C     EDITION AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
C
C     DEFINITIONS OF MOST 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROG.
C
C     SPECIAL COMMON VARIABLES:
C
C     INC$AV: STACK CONTAINING BLOCKS. EACH BLOCK CONTAINS A NETWORK'S
C             NAME, PARENT, COST, AND LIST OF CONNECTIONS.
C       NAME: NAME OF NETWORK UNDER CONSIDERATION.
C     NETLST: LIST (STACK) OF POINTERS TO TOP OF BLOCKS IN INC$AV.
C     NLSTPT: POINTS TO TOP OF STACK NETLST.
```

```
C     NTCNTR: NETWORK COUNTER - COUNTS NUMBER OF NETWORKS GENERATED SO
C             FAR.
C     NTCOST: COST OF A PARTICULAR NETWORK.
C     PARENT: NAME OF NETWORK FROM WHICH NETWORK 'NAME' WAS DERIVED.
C      $AVPT: POINTS TO FIRST FREE LOCATION IN STACK INC$AV.
C
C     HOW TO INCREASE CAPACITY OF SUBROUTINE.
C     DIMENSION: NETLST(X) - X EQUAL TO MAX NUMBER OF NETWORKS ALLOWED
C                            IN STACK + 1
C                INC$AV(Y) - Y EQUAL TO 3*X PLUS MAX TOTAL NUMBER OF
C                            CONNECTIONS IN ALL NETWORKS STORED IN STACK
C
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
      COMMON NERMAX
      COMMON    N             , M             , A             , B
     1      ,   R             , N2            , N1            , NR
     2      ,   NM            , KFLAG         , JFLAG         , COST
     3      ,   LEVM          , NRN2          , NM1           , NN2
      COMMON    ISUCC(40,40)  , LISUCC(40)    , IPRED(40,40)  , LIPRED(40)
     1      ,   INC$MX(40,40) , SUC$MX(40,40) , P$(2,1280)    , UNAME(40)
     2      ,   GLEVEL(40)    , LGLIST(40)    , HLIST(40,40)  , TIME
      COMMON    T             , RTCONN(100)   , S             , RSCONN(100)
      COMMON    IFLAG         ,POINTA         ,ESSIS(40)      ,F$1(32)
     1      ,F$UB1            ,INPTCV(32)     ,LISTC(40)      ,POINTC
     2      ,LISTL(40)        ,POINTL         ,ORIGIN(40)     ,IPATH(40)
     3      ,POINTR           ,VF$1(32)       ,VF$UB1         ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)      ,LPOTAB(40)     ,NRPLC(2)
     1      ,RPLC(2,40)       ,IDX0(32)       ,IDX0E(32)      ,IDX1(32)
     2      ,IDX1E(32)        ,SUMP(32)       ,SETT1(32)      ,NOT1
     3      ,SETS1(40)        ,NOS1           ,SETS(40)       ,NOS
     4      ,STS              ,SUMS2(32)      ,SETS2(200)     ,NOS2
     5      ,LIP              ,NODE           ,KEYA           ,KEYB
     6      ,NO0              ,NO1            ,NO1E           ,$GT
     7      ,$LTH             ,$PW            ,$NOE           ,GI
      COMMON                  NOT1SV         ,NOS1SV         ,LMTS2
      COMMON    NTCNTR        ,PARENT         ,NAME           ,INC$AV(10000)
     1      ,$AVPT            ,NETLST(500)    ,NLSTPT         ,NTCOST
      NETLST(1) = 0
      NTCNTR = 1
      NLSTPT = 1
      $AVPT = 1
      PARENT = 0
      NAME = 1
      NTCOST = COST
    1 CALL PROCCE(WORKED)
C     IF STACK EMPTY, RETURN
      IF(NETLST(NLSTPT).EQ.0)RETURN
C     CHOOSE NEW NETWORK FROM TOP OF STACK, SET UP
      DO 2 I=1,NR
      DO 2 J=1,NR
    2 INC$MX(I,J) = 0
      X = NETLST(NLSTPT)
      NLSTPT = NLSTPT - 1
      Y = NETLST(NLSTPT) + 1
      $AVPT = Y
      NAME = INC$AV(X)
      PARENT = INC$AV(X-1)
      NTCOST = INC$AV(X-2)
      X = X - 3
      DO 3 I=Y,X
      Z = INC$AV(I)
      IGATE = Z/1000                                                    E3 07650
      JGATE = Z - 1000*IGATE                                            E3 07660
    3 INC$MX(IGATE,JGATE) = 1                                           E3 07670
      CALL SUBNET                                                       E3 07680
      CALL PVALUE                                                       E3 07690
      GO TO 1                                                           E3 07700
      END                                                               E3 07710
```

```
**************************************************
*                                                *
*          THE REST OF THE SUBROUTINES           *
*          REQUIRED FOR NETTRA-E3 ARE:           *
*                                                *
*   CALS1, CONECT, FORC, MINI2, ORDRQ2, OUTPUT,  *
*         POT, RCEC, RPLCF, AND SUBNET.          *
*                                                *
*                                                *
*     (THESE ARE IDENTICAL TO SUBROUTINES OF     *
*      THE SAME NAMES LISTED FOR NETTRA-E1.)     *
*                                                *
**************************************************
```

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-75-732	2.	3. Recipient's Accession No.

4. Title and Subtitle
PROGRAM MANUAL: NOR NETWORK TRANSDUCTION BASED ON ERROR-COMPENSATION (Reference Manual of NOR Network Transduction Programs NETTRA-E1, NETTRA-E2, and NETTRA-E3)

5. Report Date
June, 1975

6.

7. Author(s)
H. C. Lai, J. N. Culliney

8. Performing Organization Rept. No.

9. Performing Organization Name and Address
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.
NSF GJ-40221

12. Sponsoring Organization Name and Address
National Science Foundation
1800 G Street, N. W.
Washington, D.C. 20550

13. Type of Report & Period Covered
Technical

14.

15. Supplementary Notes

16. Abstracts

Three NOR network transduction procedures based on error-compensation were implemented in the FORTRAN computer programs NETTRA-E1, NETTRA-E2, and NETTRA-E3. The general principles on which these programs are based are discussed in a separate report. The present report, however, describes the specific implementations of the three programs and serves as a reference manual for the program user. Preparation of input data is discussed in detail.

Transduction (transformation and reduction) procedures attempt to reduce given, non-optimal, multiple-output, multiple-level, loop-free, NOR-gate networks to "near-optimal" networks of fewer gates. The three programs described in this report, based on the sophisticated "error-compensation" concept, remove gates one at a time from the network and, after each removal, try to reconfigure the network, without employing additional gates, to compensate for any resultant errors caused in the network output(s).

17. Key Words and Document Analysis. 17a. Descriptors

Logic design, logic circuits, logical elements, programs (computers).

17b. Identifiers/Open-Ended Terms

Program manual, computer-aided-design, network transduction, network transformation, error-compensation, near-optimal networks, permissible functions, permissible functions with errors, NOR, NAND, CSPFE, CSPF.

17c. COSATI Field/Group

18. Availability Statement Release Unlimited	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 178
	20. Security Class (This Page UNCLASSIFIED	22. Price